Incremental Testing and Regression Testing

Product: SpotiBot

Inspectors: Jimmy Carlson, Janka Gal, Puja Mittal, Jason Shipp, David Worley

1. Classification of Components

1.1 Define all components

server - nginx acts as the reverse proxy server that is actually pinged by facebook and aws. The control flow goes under two routes. The user sends a https post to facebook's messenger service which is then redirected as a get to spotibot.tech/webhook which is an aws server, at which point it hits the nginx configuration which verifies the https request and encryption before proxing the request forward to pm2 to handle. The second control flow originates from our app.js where a request is handled towards spotify via https requests and responses, using nginx and pm2 in the same way as with facebook.

- Config file: (/etc/nginx/sites-available/default)
    - Listen 80 - opens port 80 for http requests
    - Listen 443 - opens port 443 for https requests
    - Proxy_pass - redirects the proper traffic towards the node server at the localhost port (8080 in our case)
    - $scheme == http - redirects all http traffic towards the https traffic, non-encrypted data is encrypted or not accepted.
- Server launch:
    - /etc/init.d/nginx reload - attempts to reload the server, fails if there are problems in the config file
    - Nginx -t - checks the config file and isolates problems in the syntax of file chains

    - Nginx -s stop - stops the server from running
- Pm2 start app
    - Launches the app itself in headless mode to be proxied by nginx

database - stores user information such that they don’t have to login repeatedly

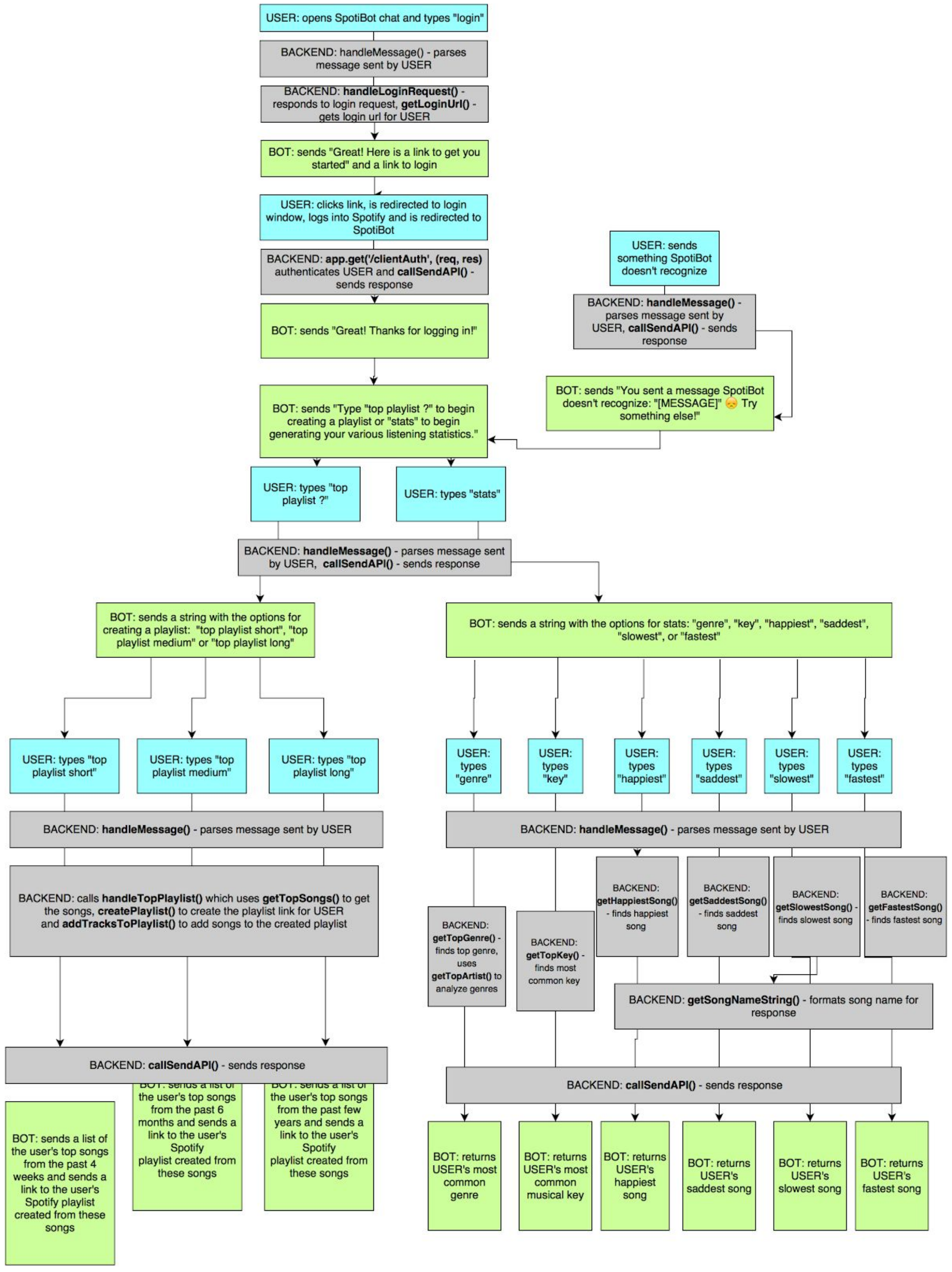
- findUser() - searches to see if the user already exists in the DB
- addUser() - adds user to DB
- updateUserAccessToken() - updates access token to a non-expired one (tokens are a part of user auth for Spotify)
- removeUser() - takes user out of the DB
- dropDB() - removes DB
- emptyCollection() - removes all entries of a DB
- isEmpty() - checks if a JSON is empty

app.js - handles backend for the bot: user authentication and communication

- handleMessage() - reads the user input and either calls functions
- handleLoginRequest() - calls node-spotify-api method which in turn generates a unique url for that user to be redirected to spotify’s login page, calls getLoginUrl as a helper method.
- getLoginUrl() - gets the login url from Spotify
- callSendAPI() - sends the response to the user from the bot
- handleTopPlaylist() - helper method to call node-spotify-api, gets the users top 50 tracks using getTopSongs, creates a new playlist with a formatted name, then passes in the songs retrieved before sending back the playlist url to the user.
- getTopSongs() - gets the list of the 50 top songs from a user in a specific time range (short - four weeks, medium - 6 months, long - a few years)
- getTopArtists() - gets the top 5 artists for the user
- createPlaylist() - creates an empty playlist in the users spotify account based upon a name passed in.
- addTracksToPlaylist() - populates a playlist with the song uri’s passed to it in an array

- getTopKey() - pulls the users top 50 tracks using getTopSongs() before
- getTopGenre() - analyzes the user’s top artists from getTopArtists() and returns the most popular genre
- getHappiestSong() - analyzes the user’s top 50 songs and returns the happiest song based on highest valence
- getSaddestSong() - analyzes the user’s top 50 songs and returns the saddest song based on lowest valence
- getFastestSong() - analyzes the user’s top 50 songs and returns the fastest song based on highest energy
- getSlowestSong() - analyzes the user’s top 50 songs and returns the slowest song based on lowest energy

Flow and User Interaction with Backend


USER: opens SpotiBot chat and types "login"
BACKEND: handleMessage() - parses message sent by USER
BACKEND: handleLoginRequest() - responds to login request, getLoginUrl() - gets login url for USER
BOT: sends "Great! Here is a link to get you started" and a link to login
USER: clicks link, is redirected to login window, logs into Spotify and is redirected to SpotiBot
BACKEND: app.get('/clientAuth', (req, res) authenticates USER and callSendAPI() - sends response
BOT: sends "Great! Thanks for logging in!"
USER: sends something SpotiBot doesn't recognize
BACKEND: handleMessage() - parses message sent by USER, callSendAPI() - sends response
BOT: sends "You sent a message SpotiBot doesn't recognize: "[MESSAGE]" 😞 Try something else!"
BOT: sends "Type "top playlist ?" to begin creating a playlist or "stats" to begin generating your various listening statistics."
USER: types "top playlist ?"
USER: types "stats"
BACKEND: handleMessage() - parses message sent by USER, callSendAPI() - sends response
BOT: sends a string with the options for creating a playlist: "top playlist short", "top playlist medium" or "top playlist long"
BOT: sends a string with the options for stats: "genre", "key", "happiest", "saddest", "slowest", or "fastest"
USER: types "top playlist short"
USER: types "top playlist medium"
USER: types "top playlist long"
BACKEND: handleMessage() - parses message sent by USER
BACKEND: calls handleTopPlaylist() which uses getTopSongs() to get the songs, createPlaylist() to create the playlist link for USER and addTracksToPlaylist() to add songs to the created playlist
BACKEND: callSendAPI() - sends response
BOT: sends a list of the user's top songs from the past 4 weeks and sends a link to the user's Spotify playlist created from these songs
BOT: sends a list of the user's top songs from the past 6 months and sends a link to the user's Spotify playlist created from these songs
BOT: sends a list of the user's top songs from the past few years and sends a link to the user's Spotify playlist created from these songs
USER: types "genre"
USER: types "key"
USER: types "happiest"
USER: types "saddest"
USER: types "slowest"
USER: types "fastest"
BACKEND: handleMessage() - parses message sent by USER
BACKEND: getTopGenre() - finds top genre, uses getTopArtist() to analyze genres
BACKEND: getTopKey() - finds most common key
BACKEND: getHappiestSong() - finds happiest song
BACKEND: getSaddestSong() - finds saddest song
BACKEND: getSlowestSong() - finds slowest song
BACKEND: getFastestSong() - finds fastest song
BACKEND: getSongNameString() - formats song name for response
BACKEND: callSendAPI() - sends response
BOT: returns USER's most common genre
BOT: returns USER's most common musical key
BOT: returns USER's happiest song
BOT: returns USER's saddest song
BOT: returns USER's slowest song
BOT: returns USER's fastest song

1.2 Which form of incremental testing did you follow

We followed bottom up incremental testing and tested our code in smaller components. As we wrote specific functions we tested them for those individual cases and then combined their functionality and expanded them, making sure the added functionality continued to work as expected. We wrote test cases as we wrote the code and used these test cases to make sure that basic functionality worked correctly before we added more complex functionality. The different functions and requests to Spotify in our application relied on the basic functionality working so bottom up incremental testing made the most sense for SpotiBot.

2. Incremental and Regression Testing

Server

Module - config file

Regression Testing

Defect No.	Description	Severity	How To Correct
1	The ssl certificate was found to be in conflict with the server name, and therefore would not be accepted by nginx reload	1	Changed the server name to correspond to www.spotibot.tech as well as spotibot.tech.
2	Facebook refused to recognize the ssl certificate as valid and signed	1	Create a certificate bundle showing the signing process as opposed to simply the self signed cert

app.js

Module - User Login

Incremental Testing

Defect No.	Description	Severity	How To Correct

1	When sending confirmation that their oauth token was received and recorded, we also sent a help message on how to get started, but the two arrived in an unpredictable order because they can get split up by facebook	3	Create a promise which chains in a timeout function which calls the second message only after half a second

Regression Testing

Defect No.	Description	Severity	How To Correct
1	Jwt tokens proved to be an ineffective and expensive way to maintain user interaction and caused hanging not accepted by Facebook	1	When users login through spotify we are give a single token to pass back towards ourselves, in which we passed the psid assigned to the user by facebook which can be used to reopen communications and tie back OAUTH tokens

Module - Communicating with User

Incremental Testing

Defect No.	Description	Severity	How To Correct
1	The user was provided with very little feedback on how they were to proceed with making a playlist	2	Provide a helper function which is accessed with a few key words minus their parameters as well as “?”

Regression Testing

Defect No.	Description	Severity	How To Correct
1		1	

Module - Create Top Playlist

Incremental Testing

Defect No.	Description	Severity	How To Correct
1	JS object properties became ill defined after promise was chained and couldn't be accessed	2	Mapped through the object and assigned them to a higher scope variable to be accessed later
2	In the date object supplied by javascript the month was perpetually incorrect	3	Months are zero indexed, nothing else is, add one

Regression Testing

Defect No.	Description	Severity	How To Correct
1	Code was being over called based upon ineffective control flow	2	Refactor the code into its own helper method which was being fed an extra variable assigned via a switch

Module - Generate Stats

Incremental Testing

Defect No.	Description	Severity	How To Correct
1	For the getHappiestSong() function, an empty variable was being returned for the happiest track.	1	Implemented a new Promise((resolve, reject) => wrapper around the logic for finding the happiest track to make sure that the logic was completed before the function returned a value.
2	The getHappiestSong() function did not return a value when implemented in the app.js file even though it worked in the testing file.	2	The error output in the server logs was analyzed and it turned out that app.js error handling required the index variable for the for loop in the function to be defined. Once it was defined, the function worked as intended.

Regression Testing

Defect No.	Description	Severity	How To Correct
1	Once the track was returned instead of an empty string, we realized the sorting function was not implemented correctly and sorted the song list with ascending values instead of descending values.	2	The sorting logic was corrected and sorted the list descending to get the song with the highest valence.

3. Update Product Backlog

#	Functional Requirements	Hours	Status	Updated Status
1	As a user, I want to be able to sign into my Facebook Messenger account	5	Sprint 1	Completed in Sprint 1
2	As a user, I want to be able to sign into my Spotify account, given that I have a Messenger account or Facebook account	5	Sprint 1	Completed in Sprint 1
3	As a user, I want to be able to tell the bot how many songs I want in the generated playlist	5	Sprint 2	Planned for Sprint 2
4	As a user, I want to be able to input a number of months and get a playlist of my top tracks from that time period	25	Sprint 1	Completed in Sprint 1
5	As a user, I want to be able to input one or more genres into the chat after prompted and get a playlist based on those genres	25	Sprint 2	Planned for Sprint 2
6	As a user, I want to be able to input one or more moods into the chat after prompted and get a playlist based on those moods	25	Sprint 2	Planned for Sprint 2
7	As a user, I want to be able to input one or more songs into the chat after prompted and get a playlist based on those songs	25	Sprint 2	Planned for Sprint 2

8	As a user, I want to be able to input one or more of my playlists into the chat after prompted and get a playlist based on those playlists	25	Sprint 2	Planned for Sprint 2
9	As a user, I want to be able to input one or more artists into the chat after prompted and get a playlist based on those artists	25	Sprint 2	Planned for Sprint 2
10	As a user, I want to be able to view what is the most common genre that I listen to.	15	Sprint 1	Completed in Sprint 1
11	As a user, I want to be able to view the most common key of the music I listen to.	15	Sprint 1	Completed in Sprint 1
12	As a user, I want to be able to see the happiest songs that I listen to.	15	Sprint 1	Completed in Sprint 1
13	As a user, I want to be able to see the saddest songs that I listen to.	15	Sprint 1	Completed in Sprint 1
14	As a user I want to be able to view the slowest songs that I listen to.	15	Sprint 1	Completed in Sprint 1
15	As a user, I want to be able to view the fastest songs that I listen to.	15	Sprint 1	Completed in Sprint 1