<u>SoundScape Test Plan and Results</u>

Part 1 – Overall Test Plan

The main plan for testing our application will be to test different components individually to ensure each portion is functioning correctly and then test the application as a whole to ensure each portion works correctly with each other. Various components we will be testing include API requests, server functionality, UI functionality, performance of the AI model, and more. We will also test how well our application performs the desired task of creating personalized playlists tailored to the user and their specific preferences at that moment. The test cases for our application are outlined below.

Part 2 – Test Case Descriptions

BF1.1 – **Basic Functionality Test 1**

BF1.2 – This test will ensure the user is able to access the published app

BF1.3 – The user will attempt to access the published app through the web by going to the public URL (still to be determined)

BF1.4 – Input: There is no input other than the user attempting to access the site

BF1.5.1 – Expected output: The user should be able to access the home screen of the app and should not encounter any errors

BF1.5.2 – Results: (Will fill in upon completion of testing)

BF1.6 – Normal

BF1.7 – Blackbox

BF1.8 – Functional

BF1.9 – Unit test

BF2.1 – **Basic Functionality Test 2**

BF2.2 – This test will ensure the user is able log into Spotify using the SoundScape app

BF2.3 – The user will access the app and then click the “Log Into Spotify” button that shows up on the screen. They will then log into Spotify and approve permissions for the app

BF2.4 – Input: The explicit input is the user providing their login information and approving permissions for the app. Implicitly, the app will also perform API requests to Spotify to retrieve a key so the user is able to use the app and the inputs to these requests will include the scopes, redirect URI, and client ID of the app

BF2.5.1 – Expected output: The user should be successfully logged into Spotify and be taken to the home screen of the app. The user should not encounter any errors

BF2.5.2 – Results: (Will fill in upon completion of testing)

BF2.6 – Normal

BF2.7 – Blackbox

BF2.8 – Functional

BF2.9 – Unit test

UI1.1 – **UI Test 1**

UI1.2 – This test will ensure the user can select and change all inputs to a desired value

UI1.3 – The tester will access the app, log into Spotify, and access the home screen. Then, they will be able to access all user inputs. They will test these by trying to enter information and making sure the UI allows them to make their inputs and does not have any errors or unintended behavior

UI1.4 – Input: The input will be the tester changing the values for all fields in the UI to their desired values. Invalid input will qualify as values that do not match the proper data type, such as string/integer/double mismatch in inputs.

UI1.5.1 – Expected output: The tester will expect that the UI will allow them to manipulate all inputs to their desired output without encountering any errors and without there being any issues with UI elements

UI1.5.2 – Results: (Will fill in upon completion of testing)

UI1.6 – Normal

UI1.7 – Blackbox

UI1.8 – Functional

UI1.9 – Unit test

UI2.1 – **UI Test 2**

UI2.2 – This test will handle validation for all fields and ensure that no inputs will accept invalid values

UI2.3 – The tester will access the app and will attempt to insert invalid inputs into each of the fields. Invalid inputs will be different for each field but will include attempting to slide a slider beyond its intended limits and inserting values beyond the intended range. Mood input selector will only accept one of the values in the dropdown as a valid selection. Discovery and tempo (BPM) sliders will only accept proper inputs. Max and min song duration will only accept proper inputs. Artist filter will only accept good data filtered in through API requests.

UI2.4 – Input: The input will be the incorrect/invalid values being inserted into each of the fields by the tester

UI2.5.1 – Expected output: The tester will expect that the UI will not allow them to insert incorrect or invalid values into each of the fields. There will either be a warning message or pop up displayed for each of the inputs

UI2.5.2 – Results: (Will fill in upon completion of testing)

UI2.6 – Abnormal/Boundary Case

UI2.7 – Blackbox

UI2.8 – Functional

UI2.9 – Unit test

UI3.1 – **UI Test 3**

UI3.2 – This test will check the functionality of the recommendations page and that selected songs are properly added to the current playlist in progress, while removed songs are properly taken out of the rotation

UI3.3 – The tester will log into the app and attempt to get song recommendations for their playlist. They will then try to both swipe songs left (discard) and right (add) and use the buttons to ensure that songs can be added or not correctly. They will also try to use the UI on the sides of the screen that allow you to make edits to the discarded or added songs.

UI3.4 – Input: The tester will provide input by adding and removing songs to the current playlist they are creating in the app

UI3.5.1 – Expected output: The tester will expect that they are able to add songs both by swiping right and by using the add button. They will also expect that they will be able to discard songs by swiping left and using the discard button. Additionally, they will expect that they can edit the added or removed songs using the UI on the sides of the screen.

UI3.5.2 – Results: (Will fill in upon completion of testing)

UI3.6 – Normal

UI3.7 – Blackbox

UI3.8 – Functional

UI3.9 – Unit test

UI4.1 – **UI Test 4**

UI4.2 – This test will ensure that all confirmations and pop-ups appear as expected

UI4.3 – The tester will log into the app and will begin interacting with inputs that will result in the UI giving a pop up or confirmation message. Some examples may include adding or discarding songs from the recommendations, fetching recommendations, and creating the playlist in Spotify

UI4.4 – Input: There will be various inputs including entering information into input fields and pressing buttons

UI4.5.1 – Expected output: The tester will expect that there will be the correct confirmation message or pop-up message for each field or button that there should be without any errors occurring

UI4.5.2 – Results: (Will fill in upon completion of testing)

UI4.6 – Normal

UI4.7 – Blackbox

UI4.8 – Functional

UI4.9 – Unit test

UI4.1 – **UI Test 5**

UI5.2 – This test will ensure that all the app is able to be used on a phone in the same way it can be used on a computer

UI5.3 – The tester will navigate to the URL of the app using their phone/mobile device instead of their computer. They will log in and attempt all functionalities of the app, ensuring that there are no cases where the UI is unusable or where it does not adapt properly to the smaller screen size

UI5.4 – Input: There is no input for this test except for the tester navigating to the app using their phone

UI5.5.1 – Expected output: The UI will appear “good-looking” on the mobile version of the app and will not have any components or controls that are difficult to use or that are not available on the smaller screen

UI5.5.2 – Results: (Will fill in upon completion of testing)

UI5.6 – Normal

UI5.7 – Blackbox

UI5.8 – Functional

UI5.9 – Unit test

ST1.1 – **Server Test 1**

ST1.2 – This test will test that the node server is functional upon deployment

ST1.3 – The team will attempt to communicate to the deployed node server using dummy API requests through Postman. This test is purely to ensure that the sever is functioning after deployment

ST1.4 – Input: Dummy data in JSON format that will result in an expected 200 OK response from the server

ST1.5.1 – Expected output: We expect the server to return a response of 200 OK and return the correct data required

ST1.5.2 – Results: (Will fill in upon completion of testing)

ST1.6 – Normal

ST1.7 – Whitebox

ST1.8 – Functional

ST1.9 – Unit test

ST2.1 – **Server Test 2**

ST2.2 – This test will ensure the server that handles the AI model is functional after deployment

ST2.3 – To test the server handling the AI model, the team will make an API request to that server using dummy data inside Postman. This test is purely to ensure that the sever is functioning after deployment

ST2.4 – Input: Dummy data that will be used in an API request inside Postman that will result in a correct response from the server (the exact format is not known at this time as the server and AI model are still in development)

ST2.5.1 – Expected output: We expect the server to return a correct response after the API request is made and any required data will be returned (exact specifications unknown at this time)

ST2.5.2 – Results: (Will fill in upon completion of testing)

ST2.6 – Normal

ST2.7 – Whitebox

ST2.8 – Functional

ST2.9 – Unit test

ST3.1 – **Server Test 3**

ST3.2 – This test will ensure the frontend can communicate with the servers and all API requests are functioning correctly

ST3.3 – The team will test various API requests through the frontend of the app to ensure all areas of the app that make server requests are functioning correctly. First, we will log into Spotify and confirm that functions correctly. Then, we will individually test each component that requires requests to the backend (such as fetching weather data, getting music data, and others that may be implemented at time of testing)

ST3.4 – Input: The input will be the data of the team member testing each component and interacting with the frontend to trigger backend requests

ST3.5.1 – Expected output: We expect that all requests to the backend will succeed and the requested information will be returned in the correct format without any errors or unexpected behavior

ST3.5.2 – Results: (Will fill in upon completion of testing)

ST3.6 – Normal

ST3.7 – Blackbox

ST3.8 – Functional

ST3.9 – Integration

AI1.1 – **AI Test 1**

AI1.2 – This test will make sure the AI model can be accessed from the frontend after deployment

AI1.3 – The user will go to the deployed site, log into the app and Spotify, and attempt to get music recommendations from the AI model using the UI within the app

AI1.4 – Input: The input will be any user data that is sent to the AI model as well as the user interacting with the UI and any values associated with that

AI1.5.1 – Expected output: We expect that song recommendations will appear inside the SoundScape app from the AI model after this test is completed

AI1.5.2 – Results: (Will fill in upon completion of testing)

AI1.6 – Normal

AI1.7 – Blackbox

AI1.8 – Functional

AI1.9 – Unit test

AI2.1 – **AI Test 2**

AI2.2 – This test will ensure that the recommendations given by the AI model are accurate to the data provided and deemed acceptable by the tester

AI2.3 – The user will access the app and will specify any of the user inputs that they would like to tune the playlist to their liking (note: the tester will attempt this test multiple times with different user inputs to ensure all are functioning correctly). Then, they will attempt to get music recommendations from the AI model

AI2.4 – Input: All UI inputs that the user wishes to use to tune the music to their liking, as well as any user data that is automatically collected (such as location, weather, and listening history)

AI2.5.1 – Expected output: We expect that the user will receive song recommendations that fit all of the custom parameters they specified in the UI. Additionally, these recommendations should be considered “good and accurate” by the tester

AI2.5.2 – Results: (Will fill in upon completion of testing)

AI2.6 – Normal

AI2.7 – Blackbox

AI2.8 – Performance

AI 2.9 – Unit test

O1.1 – **Output Test 1**

O1.2 – This test will ensure that the user is able to export their created playlists to Spotify successfully

O1.3 – The tester will log into the app, get song recommendations, and create a playlist of their choice using the controls in the app. They will then hit the “export to Spotify” button within the app. They will then go to the Spotify app and confirm that the playlist has been created in their account

O1.4 – Input: The custom playlist that the tester creates within the SoundScape app

O1.5.1 – Expected output: The tester will expect that their custom playlist will be created in Spotify, and they will be able to listen to the songs there

O1.5.2 – Results: (Will fill in upon completion of testing)

O1.6 – Normal

O1.7 – Blackbox

O1.8 – Functional

O1.9 – Integration

Part 3 – Test Case Matrix

Test Code	**Normal / Abnormal / Boundary**	**Blackbox / Whitebox**	**Functional / Performance**	**Unit / Integration**
BF1	Normal	Backbox	Functional	Unit

BF2	Normal	Blackbox	Functional	Unit
UI1	Normal	Blackbox	Functional	Unit
UI2	Abnormal / Boundary	Blackbox	Functional	Unit
UI3	Normal	Blackbox	Functional	Unit
UI4	Normal	Blackbox	Functional	Unit
UI5	Normal	Blackbox	Functional	Unit
ST1	Normal	Whitebox	Functional	Unit
ST2	Normal	Whitebox	Functional	Unit
ST3	Normal	Blackbox	Functional	Integration
AI1	Normal	Blackbox	Functional	Unit
AI2	Normal	Blackbox	Performance	Unit
O1	Normal	Blackbox	Functional	Integration