Software Test Plan

Document

for

EarWorm

Version 1.0
17 December 2022

Team: Keyboard Buddies

Members: Alvaro Lopez-Romero, Mitchell Mercer,
Nathalie Claire Tivar, Rodolfo Rivera, Charles Quebral

Supervisor: Edmund Dantes
Computer Science Department
California State University, Northridge

Revision History

Version	By	Change Description	Date
1.0	All team members	Original Document	12/17/2022
1.1			
1.2			
1.3			

1.0 INTRODUCTION

EarWorm, a senior capstone project, seeks to connect individuals and artists with like minded individuals based on criteria such as music taste, sex, age, and location. EarWorm is a web-based application that will allow users to create their accounts and synchronize with their existing Spotify accounts. All data will be saved in a secured Google Cloud database. The EarWorm application will utilize a project team created API client to make API calls to Spotify's API for user data. The data will be fed to the MatchMaker class, which will be trained by Machine Learning, to give the user nearby matches on defined criteria.

2.0 OBJECTIVES AND TASKS

2.1 Objectives

The objective of the test plan is to verify that the functionality of EarWorm works according to the specification. This document lays out how testing will function and progress.

The test plan will execute and verify different classes and their methods to identify, fix, and retest all severity level defects.

The final product of the EarWorm test should:

- Allow new user creation
- Synchronize that user's EarWorm account with their Spotify account.
- Store their unique account information securely in the database.
- Allow the user to see other users in the application that are nearby and match their music taste.

2.2 Tasks

- Each class will have unit testing performed before being deployed into the repository for integration testing.
- Build a test database to run the full system against.
- Validate user creation in the database.
- During the integration testing EarWorm will contain pre-loaded test data.
- System testing with Alpha test users.
- Acceptance testing with Beta test users.
- Problem reporting through Peer Review Artifact document.

3.0 SCOPE

General

Testing will ensure that our designed functionalities work as intended. Testing unique user data will ensure that valid matches are made for users with similar music tastes.

- User creation
- Unique user ID, no repetitive users
- Database content and integration
- Integration with Spotify API
- MatchMaker classification for matching users

4.0 TESTING STRATEGY

The strategy we will approach testing will be one of trying to test the absolute extreme of music taste prediction and end user experience. This will be accomplished by feeding the Machine Learning module extreme, fabricated, edge cases. Additionally fake user profiles will be generated with deliberately incompatible traits and will then be observed for potential error.

4.1 Unit Testing Definition:

The minimum degree of comprehensiveness for testing will be developer generated failures based on the points of:

- Comparing deliberately false music taste generation with “accurate” results
- Deliberately incompatible music taste matching
- Deliberate attempting population of the database with data incompatible with user generation.

Participants:

The entire team will participate as well as invited beta testers. Including but not limited to CSUN students and staff.

Methodology:

The entire team will participate in writing test scripts. Generated failures will be observed and documented using these scripts, while bug reports will be received from invited beta testers and parsed by all developers.

4.2 System and Integration Testing Definition:

The two main components that require integration testing are the Users matching and Music Taste assignment modules.

Participants:

Alvaro Lopez-Romero, Mitchell Mercer, Nathalie Claire Tivar, Rodolfo Rivera, Charles Quebral will be responsible for integration testing.

Methodology:

Modules will be individually tested and certified. After which modules will be fully supervised while generating results. In the second stage software will be semi supervised. Finally completely unsupervised testing will occur with any errors only being reported by end users.

4.3 Performance and Stress

N/A

5.0 ENVIRONMENT REQUIREMENTS (TOOLS)

At current the application must be accessible via modern web browser. Project viability will be tested using machines utilizing Windows 10/11 and MacOS paired with operating systems Chrome, Safari, and Firefox. Edge cases like accessing the website using Operating Systems like Linux and iOS Safari and other mobile Operating Systems and web browsers will also be explored.

Additionally the Spring Boot Framework furnishing tools for testing and VSCode will be utilized to test source Java Code.

6.0 TEST SCHEDULE

2/24 – Java Api and Source Code tested and verified
3/15 – Machine Learning Model Verified
3/31 – Initial System Integration finalized and completed
4/15 – Semi Supervised Testing Begins
5/01 – Final Testing and Tweaking Begins

7.0 CONTROL PROCEDURES

Problem Reporting

Upon an observed failure a crash report will be written, included will be the specific details of the either

A.) The users spotify accounts and report music taste.
B.) The Two matched user profiles.
C.) The generated music taste and all inputs that resulted in invalid output.

Change Requests

Date	Requestee	Observed Where	Error	Suggested Solution

8.0 RESOURCES/ROLES & RESPONSIBILITIES

Team Member	Roles	Email
Alvaro Lopez-Romero	Team Lead / Developer	alvaro.lopez-romero.317@my.csun.edu
Mitchell Mercer	Developer / Tester	mitchell.mercer.062@my.csun.edu
Nathalie Claire Tivar	Front End Developer / Tester	nathalie-claire.tivar.562@my.csun.edu
Rodolfo Rivera	Front End/Back End Developer / Tester	rodolfo.rivera.712@my.csun.edu
Charles Quebral	Back End Developer / Tester	charles.quebral.447@my.csun.edu

9.0 RISKS/ASSUMPTIONS

Ongoing