Note: Our project follows a different path and thus we don't have a design for a project yet, instead this document will be updated as a design is created and instead be focused around our design process and potential designs. For more information please contact our client and advisor, Rachel Shannon.

Potential Design: Monitor brain waves while listening to music then generate art from those. This project will be made in Java.

5 Testing

*Testing is an **extremely** important component of most projects, whether it involves a circuit, a process, power system, or software.*

The testing plan should connect the requirements and the design to the adopted test strategy and instruments. In this overarching introduction, given an overview of the testing strategy and your team's overall testing philosophy. Emphasize any unique challenges to testing for your system/design.

In the sections below, describe specific methods for testing. You may include additional types of testing, if applicable to your design. If a particular type of testing is not applicable to your project, you must justify why you are not including it.

When writing your testing planning consider a few guidelines:

- *Is our testing plan unique to our project? (It should be)*
- *Are you testing related to all requirements? For requirements you're not testing (e.g., cost related requirements) can you justify their exclusion?*
- *Is your testing plan comprehensive?*
- *When should you be testing? (In most cases, it's early and often, not at the end of the project)*

5.1 Unit Testing

Unit	How Unit is Tested
Brain Wave Monitoring	Unit will be giving faked data, and we will test if the appropriate output is recorded.
Music Player	Unit will give unit information from an external source and play without error.
Brain Wave Processor	Unit will be supplied with information from the processor and tested to see if it can correctly translate the data.

AI Art Generator	We will be giving fake data, and we will observe if this unit can display art after given data without any errors.
Display	We will give unit mocked dependencies and assert that the unit can display data without errors.

Tools To Be Used

We will use JUnit, AssertJ, and Mockito. We will be following best practices and using TDD to create the unit test.

5.2 Interface Testing

Interface	**How the Interface will be Tested**
Audio Input/Output	We will use an external tool (oscilloscope) to analyze audio output to assert that audio is generated successfully. We can also use waveform generators to test dummy sounds before audio if fully implemented and needed.
Brain Wave Monitor	We will use Fast Fourier transform (FFT) to break apart the brain wave in order to assert that the brain wave monitor contains the appropriate components of a brain wave. We can also use spectrograms.
Display Unit	We will use a test framework to assert graphics are drawn successfully.

We will need to test electrical components with equipment available in the lab, such as an LCR meter, oscilloscope, and multimeter.

5.3 Integration Testing

The critical integration paths include:

- Reading the brain waves to the brain wave processor to the AI Art Generator to the Display Unit
- Audio Input to the brain wave monitor to the AI Art Generator to the display unit

These critical paths will be tested using end-to-end tests, using mocked brain wave data and oscilloscope and a graphic test framework on the other.

5.4 System Testing

The very first step would be to create some sort of test plan. From there, we would create system test cases and test scripts, and then we would generate the testing data required for the tests. Next, we would execute those test cases and script and report the bugs if there are any. After that, we would complete black box testing and see if our integration tests are able to all work properly and input and output the information we are looking for.

5.5 Regression Testing

Because our code will be created using test-driven development, we will be assured that new features do not break old features with these tests. These tests will be run whenever code is pushed to the repository using a CI/CD Pipeline. All new code will be committed this way, ensuring the code is not breaking current features.

5.6 Acceptance Testing

By cycling constant feedback with our client, we can be assured that our design matches our requirements. To ensure that these requirements are working as expected, we will utilize test users who will give us a variety of sample data to test our system with. Our client will be involved with these tests to approve everything as expected.

5.7 Results

The results of our testing will assure us that our design will work as expected, and no matter what features we add, they will not break previous features. If any results indicate bugs or room for improvement, we will not move on to higher level testing until we ensure we have functional modules or components to integrate. It will also ensure the safety and privacy of any users.