Dissonance - Innovation Track Action Plan

1. Context

Project Background

Dissonance is a software solution designed to alter audio files in a way that remains imperceptible to human listeners but disrupts AI-driven analysis and recognition. Inspired by projects like Nightshade and Fawkes, which protect visual and facial data from AI models, Dissonance aims to provide similar protection for audio content.

Problem Identification & Solution

With the rise of AI-powered audio analysis tools, there is an increasing risk of unauthorized data mining, surveillance, and misuse of audio recordings. Musicians, content creators, and privacy-conscious users need a way to protect their work and personal data from being exploited by AI models.

Dissonance addresses this issue by generating adversarial perturbations that subtly modify the acoustic properties of an audio file. These modifications remain undetectable to the human ear but cause AI models to misinterpret, distort, or completely fail in recognizing the content.

Project Goals

- Develop a tool that applies adversarial perturbations to audio files to disrupt AI-based recognition.
- Ensure that the modified audio remains indistinguishable to human listeners.

- Create an intuitive user interface for easy adoption by musicians, content creators, and security-conscious users.
- Validate effectiveness through AI model testing and continuous improvement.

2. Technical Specifications

Technology Stack

Front-End Technologies:

- **JUCE** – Chosen for developing the desktop application UI, providing a robust framework for audio processing and cross-platform compatibility.
- **React.js** – Used for building the website where users can download the software.
- **HTML, CSS, JavaScript** – Core web technologies for UI structuring and styling.
- **Figma** – Used for designing and prototyping the user interface.

Back-End Technologies:

- **Django (Python)** – Chosen for its robust framework and seamless API integration.
- **C++** – Used for performance-critical audio processing.
- **RtAudio** – A real-time audio processing library essential for modifying audio signals.

These technologies were selected to balance ease of development, scalability, and high-performance audio processing.

Key Features & User Stories

Adversarial Perturbation Generation

- **User Story:** *As a musician, I want to protect my original compositions from AI-based plagiarism detection, so I can prevent unauthorized use of my work.*
- **User Story:** *As a journalist, I want to ensure my recorded interviews are not used for unauthorized AI-based voice synthesis, so I can maintain confidentiality and integrity.*
- **User Story:** *As a podcaster, I want to apply AI-disrupting modifications to my episodes while preserving sound quality, so my content remains exclusive and protected.*

Customizable Protection Levels

- **User Story:** *As a security-conscious user, I want to choose different levels of AI resistance (e.g., light, moderate, strong perturbation) so I can tailor the protection to my needs.*
- **User Story:** *As an audio engineer, I want to experiment with different perturbation levels to find the best balance between security and audio fidelity.*

Real-time Preview

- **User Story:** *As a user, I want to compare the original and protected audio through a playback feature so I can ensure the modifications do not affect my listening experience.*
- **User Story:** *As a musician, I want to instantly preview the impact of AI protection on different instruments in my track so I can make informed decisions.*

AI Resistance Testing

- **User Story:** *As a developer, I want to evaluate how AI models interpret protected audio files so I can improve effectiveness and ensure long-term viability.*
- **User Story:** *As a privacy advocate, I want to test my modified audio against popular AI recognition tools so I can verify its security.*

Technical Direction & Execution Plan

Sprint Roadmap (12-month plan)

Sprint 1–2 (Weeks 1–6): Research & PoC

- Implement a minimal CLI adversarial engine.
- Evaluate Whisper, DeepSpeech for attack viability.

Sprint 3–5 (Weeks 7–12): UI/UX Prototyping

- Develop Figma mockups.
- Build early JUCE prototype.
- Begin frontend/backend software.

Sprint 6–8 (Weeks 13–18): MVP Build

- Integrate after render preview.

- Finalize perturbation level controls.
- Launch internal MVP with Django backend.

Sprint 9–11 (Weeks 19–28): AI Resistance Evaluation

- Automate AI model testing suite.
- Develop scoring for resistance/fidelity tradeoffs.
- Begin optimization of perturbation engine.

Sprint 12–14 (Weeks 29–38): Community Beta

- Launch private beta on Discord.
- Collect user feedback.
- Patch issues and improve usability.

Sprint 15–18 (Weeks 39–52): Optimization & Release

- Performance tuning in C++.
- Begin plugin (VST/AU) feasibility research.
- Launch public release.

Work Organization & Roles

Since the team consists of a maximum of 4 members, roles will be shared and adjusted based on skills and workload. A flexible, collaborative approach will be maintained throughout the development cycle.

Area	Responsibilities
Project & Research Lead	Oversees roadmap, coordinates AI research, and ensures project alignment
Audio Development	Handles C++ engine, JUCE integration, real-time processing
Backend & Testing	Builds Django API, develops AI resistance testing suite
UI/UX & Frontend	Designs Figma prototypes, implements JUCE and React UI, handles user testing

Additional responsibilities like community engagement, documentation, and QA will rotate or be shared depending on the phase.

Tooling: Notion for task tracking, GitHub for code + CI/CD, Discord for user engagement.

3. Non-Technical Specifications

Scalability & Performance

- Optimize the adversarial perturbation algorithm for efficiency.
- Ensure an audio preview.
- Design a modular architecture to facilitate future improvements.

Security & Compliance

- Implement encryption to protect user data.
- Adhere to data privacy regulations (GDPR, CCPA) to ensure legal compliance.
- Research and apply the latest advancements in adversarial machine learning for improved effectiveness.

User Experience & Accessibility

- Develop a streamlined UI for non-technical users.
- Provide detailed documentation and user guides.
- Ensure accessibility features for a diverse user base.

Collaboration & Community Engagement

- Engage with researchers and industry professionals in AI.
- Establish a community to receive feedback.
- Potentially advance towards common plugin formats (VST, AU, AAX, etc...).
- Maintain an active Discord server for discussions and idea-sharing. (Invite code: QdKd8WKG8V)
- Open a GitHub Discussions page to gather contributions and suggestions.

Technology Evaluation & Protection

- Emerging Technologies to Benchmark: AI models capable of recreating music by listening to it.
- Testing Methodology: The system will provide modified audio to an AI model, then prompt it to define the genre and recreate similar music. If the AI's output closely resembles the original, the perturbations must be adjusted.
- Technology Protection: Dissonance will be closed source to slow down reverse engineering of the adversarial perturbation algorithm.
- Safeguarding Strategy: The algorithm will undergo continuous adaptation since AI recognition models evolve over time.

Monetization & Business Model

- Explore freemium and enterprise licensing models.
- Develop strategic partnerships with content creators and digital rights organizations.

Next Steps

- Finalize the first prototype and begin testing.
- Establish early partnerships and gather feedback.
- Prepare for beta launch and user testing.