Contributor Onboarding Guide

Welcome to the Audityzer Community!

Thank you for your interest in contributing to Audityzer! This guide will help you get started as a contributor and make your first meaningful contribution to our open-source audio analysis project.

Table of Contents

Getting Started

Prerequisites

- **Git**: Version control system
- **Node.js**: v16+ for JavaScript development
- **Python**: v3.8+ for Python components
- **Basic Audio Knowledge**: Understanding of audio concepts helpful but not required

First Steps

1. **Star the Repository**
   - Visit github.com/audityzer/audityzer (https://github.com/audityzer/audityzer)
   - Click the “Star” button to show your support

2. **Join Our Community**
   - Discord: discord.gg/audityzer (https://discord.gg/audityzer)
   - Introduce yourself in #introductions
   - Ask questions in #help

3. **Read the Documentation**
   - Project README (https://github.com/audityzer/audityzer/blob/main/README.md)
   - API Documentation (https://docs.audityzer.com)
   - Code of Conduct (https://github.com/audityzer/audityzer/blob/main/CODE_OF_CONDUCT.md)

Development Environment Setup

1. Fork and Clone

```
# Fork the repository on GitHub, then clone your fork
git clone https://github.com/YOUR_USERNAME/audityzer.git
cd audityzer

# Add upstream remote
git remote add upstream https://github.com/audityzer/audityzer.git
```

2. Install Dependencies

```
# Install Node.js dependencies
npm install

# Install Python dependencies (if working on Python components)
pip install -r requirements.txt

# Install development tools
npm install -g eslint prettier jest
```

3. Environment Configuration

```
# Copy environment template
cp .env.example .env

# Edit .env with your local settings
# Most defaults work for development
```

4. Verify Installation

```
# Run tests to ensure everything works
npm test

# Start development server
npm run dev

# Try the CLI
npm run cli -- --help
```

5. IDE Setup (Recommended)

VS Code Extensions

- **ESLint**: Code linting
- **Prettier**: Code formatting
- **Jest**: Test runner integration
- **GitLens**: Enhanced Git integration
- **Audio Waveform**: Audio file visualization

Configuration Files

The repository includes:

- `.vscode/settings.json` - VS Code workspace settings
- `.eslintrc.js` - ESLint configuration
- `.prettierrc` - Prettier formatting rules
- `jest.config.js` - Jest testing configuration

Understanding the Codebase

Project Structure

```
audityzer/
├── src/                    # Source code
│   ├── core/              # Core audio analysis algorithms
│   ├── cli/               # Command-line interface
│   ├── api/               # REST API server
│   ├── plugins/           # Plugin system
│   └── utils/             # Utility functions
├── tests/                 # Test files
├── docs/                  # Documentation
├── examples/              # Usage examples
├── scripts/               # Build and deployment scripts
└── playground/            # Interactive web playground
```

Key Components

Core Audio Analysis (`src/core/`)

- **`analyzer.js`** : Main analysis engine
- **`features/`** : Feature extraction algorithms (MFCC, spectral, etc.)
- **`formats/`** : Audio format handling
- **`processing/`** : Signal processing utilities

CLI Interface (`src/cli/`)

- **`index.js`** : Main CLI entry point
- **`commands/`** : Individual CLI commands
- **`config/`** : Configuration management

Plugin System (`src/plugins/`)

- **`manager.js`** : Plugin loading and management
- **`api.js`** : Plugin API definitions
- **`builtin/`** : Built-in plugins

Architecture Overview

```
┌─────────────────┐    ┌─────────────────┐    ┌─────────────────┐
│   Input Layer   │───▶│  Analysis Core  │───▶│  Output Layer   │
│                 │    │                 │    │                 │
│ • File I/O      │    │ • Feature       │    │ • JSON Export   │
│ • Stream Input  │    │   Extraction    │    │ • Visualization │
│ • Format Conv.  │    │ • Signal Proc.  │    │ • API Response  │
└─────────────────┘    └─────────────────┘    └─────────────────┘
         │                      │                      │
         ▼                      ▼                      ▼
┌─────────────────┐    ┌─────────────────┐    ┌─────────────────┐
│  Plugin System  │    │  Configuration  │    │   Monitoring    │
│                 │    │                 │    │                 │
│ • Custom        │    │ • YAML/JSON     │    │ • Performance   │
│   Extractors    │    │ • Environment   │    │ • Error Tracking│
│ • Third-party   │    │ • CLI Args      │    │ • Usage Stats   │
│   Integrations  │    │ • GUI Settings  │    │ • Health Checks │
└─────────────────┘    └─────────────────┘    └─────────────────┘
```

Contribution Workflow

1. Choose an Issue

- Browse Good First Issues (https://github.com/audityzer/audityzer/labels/good%20first%20issue)
- Check Help Wanted (https://github.com/audityzer/audityzer/labels/help%20wanted) labels
- Propose new features in Discussions (https://github.com/audityzer/audityzer/discussions)

2. Create a Branch

```
# Update your fork
git checkout main
git pull upstream main

# Create feature branch
git checkout -b feature/your-feature-name
# or
git checkout -b fix/issue-number-description
```

3. Make Changes

- Follow our coding standards
- Write tests for new functionality
- Update documentation as needed
- Commit frequently with clear messages

4. Test Your Changes

```
# Run all tests
npm test

# Run specific test suites
npm run test:unit
npm run test:integration
npm run test:e2e

# Check code coverage
npm run test:coverage

# Lint your code
npm run lint

# Format your code
npm run format
```

5. Submit Pull Request

```
# Push your branch
git push origin feature/your-feature-name

# Create pull request on GitHub
# Use the PR template and fill in all sections
```

Pull Request Guidelines

- **Title**: Clear, descriptive title
- **Description**: Explain what and why
- **Testing**: Describe how you tested
- **Screenshots**: Include for UI changes
- **Breaking Changes**: Clearly document any breaking changes
- **Checklist**: Complete the PR template checklist

Types of Contributions

Bug Fixes

- Fix reported issues
- Improve error handling
- Resolve edge cases
- Performance optimizations

Getting Started:

1. Look for issues labeled `bug`
2. Reproduce the issue locally
3. Write a test that fails
4. Fix the bug
5. Ensure the test passes

New Features

- Audio analysis algorithms
- CLI commands
- API endpoints
- Plugin integrations

Process:
1. Discuss in GitHub Discussions first
2. Create detailed design document
3. Get approval from maintainers
4. Implement with tests
5. Update documentation

Documentation

- API documentation
- Tutorials and guides
- Code comments
- README improvements

Areas Needing Help:
- Beginner tutorials
- Advanced use cases
- API examples
- Video tutorials

Testing

- Unit tests
- Integration tests
- Performance benchmarks
- Cross-platform testing

Testing Priorities:
- Audio format compatibility
- Cross-platform functionality
- Performance regression tests
- Edge case handling

UI/UX Improvements

- Playground interface
- CLI user experience
- Documentation website
- Error messages

Plugin Development

- Third-party integrations
- Custom feature extractors
- Output format handlers
- Visualization tools

Coding Standards

JavaScript/Node.js

```
// Use ES6+ features
const { analyze } = require('./analyzer');

// Async/await preferred over callbacks
async function processAudio(filePath) {
  try {
    const result = await analyze(filePath);
    return result;
  } catch (error) {
    logger.error('Analysis failed:', error);
    throw error;
  }
}

// Clear variable names
const audioFeatures = extractFeatures(audioData);
const spectralCentroid = calculateSpectralCentroid(spectrum);

// JSDoc comments for public APIs
/**
 * Extracts MFCC features from audio data
 * @param {Float32Array} audioData - Raw audio samples
 * @param {Object} options - Configuration options
 * @param {number} options.sampleRate - Audio sample rate
 * @param {number} options.numCoeffs - Number of MFCC coefficients
 * @returns {Promise<number[]>} MFCC coefficients
 */
async function extractMFCC(audioData, options = {}) {
  // Implementation
}
```

Python

```python
"""Audio analysis utilities."""

import numpy as np
from typing import List, Optional, Dict, Any

def extract_mfcc(
    audio_data: np.ndarray,
    sample_rate: int,
    num_coeffs: int = 13
) -> np.ndarray:
    """Extract MFCC features from audio data.

    Args:
        audio_data: Raw audio samples
        sample_rate: Audio sample rate in Hz
        num_coeffs: Number of MFCC coefficients to extract

    Returns:
        MFCC coefficients as numpy array

    Raises:
        ValueError: If audio_data is empty or invalid
    """
    if len(audio_data) == 0:
        raise ValueError("Audio data cannot be empty")

    # Implementation
    return mfcc_coefficients
```

Commit Messages

Follow Conventional Commits (https://www.conventionalcommits.org/):

```
type(scope): description

feat(core): add real-time MFCC extraction
fix(cli): resolve Windows path handling issue
docs(api): update authentication examples
test(core): add spectral analysis unit tests
refactor(plugins): simplify plugin loading logic
```

Code Review Process

1. **Automated Checks**: CI/CD runs tests and linting
2. **Peer Review**: At least one maintainer reviews
3. **Testing**: Reviewer tests functionality
4. **Documentation**: Ensure docs are updated
5. **Approval**: Maintainer approves and merges

Community Guidelines

Code of Conduct

We follow the Contributor Covenant (https://www.contributor-covenant.org/):
- **Be respectful** and inclusive
- **Be collaborative** and constructive
- **Be patient** with newcomers
- **Be professional** in all interactions

Communication Channels

Discord Server

- **#general**: General discussion
- **#development**: Development-related topics
- **#help**: Get help with issues
- **#showcase**: Share your projects
- **#contributors**: Contributor-only channel

GitHub

- **Issues**: Bug reports and feature requests
- **Discussions**: Design discussions and questions
- **Pull Requests**: Code contributions
- **Wiki**: Community-maintained documentation

Best Practices

- **Search first**: Check if your question has been asked
- **Be specific**: Provide details and context
- **Share code**: Use code blocks and examples
- **Follow up**: Update on progress and solutions
- **Help others**: Answer questions when you can

Recognition and Rewards

Contributor Levels

Newcomer

- First contribution merged
- Welcome package and Discord role
- Mentioned in monthly newsletter

Regular Contributor

- 5+ contributions merged
- Contributor badge on GitHub profile
- Access to contributor-only Discord channel
- Early access to new features

Core Contributor

- 20+ contributions or significant impact
- Listed on project website

- Invited to monthly maintainer meetings
- Conference speaking opportunities

Maintainer

- Trusted with repository access
- Decision-making authority
- Mentorship responsibilities
- Recognition at conferences

Recognition Programs

Monthly Spotlights

- Featured in newsletter
- Social media recognition
- Discord announcement
- GitHub profile highlight

Annual Awards

- **Outstanding Contributor**: Most impactful contributions
- **Community Champion**: Best community support
- **Innovation Award**: Most creative solutions
- **Mentor of the Year**: Best newcomer support

Swag and Rewards

- Audityzer stickers and t-shirts
- Conference ticket sponsorship
- Professional development opportunities
- Letter of recommendation for job applications

Getting Help

When You’re Stuck

1. **Check Documentation**: Often answers are already available
2. **Search Issues**: Someone might have faced the same problem
3. **Ask in Discord**: Real-time help from community
4. **Create GitHub Issue**: For bugs or feature requests
5. **Join Office Hours**: Weekly Q&A with maintainers

Mentorship Program

- **Buddy System**: Paired with experienced contributor
- **Code Review**: Detailed feedback on contributions
- **Career Guidance**: Advice on open-source career development
- **Project Planning**: Help choosing appropriate contributions

Resources

- **Documentation**: docs.audityzer.com (https://docs.audityzer.com)
- **Examples**: github.com/audityzer/examples (https://github.com/audityzer/examples)
- **Tutorials**: youtube.com/audityzer (https://youtube.com/audityzer)
- **Blog**: audityzer.com/blog (https://audityzer.com/blog)

Office Hours Schedule

- **Tuesdays 2:00 PM UTC**: General Q&A
- **Thursdays 6:00 PM UTC**: Technical deep dives
- **Saturdays 10:00 AM UTC**: Newcomer-friendly session

Next Steps

Your First Contribution

1. **Join Discord** and introduce yourself
2. **Set up development environment**
3. **Pick a "good first issue"**
4. **Ask questions** if you need help
5. **Submit your first PR**

Ongoing Involvement

- **Regular contributions**: Aim for monthly contributions
- **Community participation**: Help others in Discord
- **Feature proposals**: Suggest improvements
- **Documentation**: Help improve guides and tutorials
- **Testing**: Try new features and report issues

Long-term Growth

- **Specialize**: Become expert in specific area
- **Mentor**: Help onboard new contributors
- **Lead**: Take ownership of features or components
- **Speak**: Present at conferences and meetups
- **Maintain**: Join the core maintainer team

Welcome to the Team!

Contributing to open source is a journey, not a destination. Every contribution, no matter how small, makes a difference. We're excited to have you join the Audityzer community and look forward to seeing what amazing things we'll build together!

Questions? Don't hesitate to reach out:
- Discord: discord.gg/audityzer (https://discord.gg/audityzer)
- Email: contributors@audityzer.com
- GitHub: @audityzer (https://github.com/audityzer)

Happy coding!