**VibeScope – Ambient Emotion Radar**

**High Level Design Document**

**Djuvane Martin**

CSMS Degree Program  
Full Sail University  
June 9, 2025

**Table of Contents**

| **Section** | **Page** |
| --- | --- |
| **Title Page** | **1** |
| **Table of Contents** | **2** |
| **Introduction** | **3** |
| **Project Concept / Pitch** | **3** |
| **Technical Design Goals** | **3** |
| **Development APIs / Programming Languages** | **3** |
| **Development Standards / Cold-Start Procedures** | **3** |
| **Project Functionality** | **4** |
| **User Modes** | **4** |
| **Main Mechanics / Sub Mechanics** | **4** |
| **User Interface / Inputs / Sensors** | **4** |
| **Graphics / Animation / Video** | **4** |
| **Audio** | **4** |
| **Networking** | **4** |
| **Databases / Server-Side Technology** | **4** |
| **Localization Plan** | **4** |
| **Custom Topics** | **5** |
| **Additional Technical Information** | **5** |
| **Mockups / Flowcharts** | **6** |
| **Production Documentation** | **7** |
| **Feature List** | **7** |
| **Master Task List** | **7** |
| **Development Roadmap** | **8** |
| **Verification and Validation (V&V)** | **9** |
| **References** | **10** |

**Introduction**

**Project Concept / Pitch**

A real-time desktop app that listens to ambient audio and visualizes the mood of the environment through color-based feedback.

**Technical Design Goal(s)**

The main technical goal is to develop a modular mood classification system that analyzes live audio input and classifies it using interchangeable logic strategies. The project also aims to display mood-based visuals in real time and support easy extension with new analysis methods (e.g., machine learning). Completing the entire development lifecycle — from design to testing — is also a personal milestone.

**Development APIs / Programming Languages**

* **Language:** Python
* **Audio Input:** sounddevice, pyaudio
* **Signal Processing:** numpy, scipy
* **User Interface / Visualization:** pygame
* **Testing:** unittest
* **Diagrams:** PlantUML, draw.io

**Development Standards / Cold-Start Procedures**

* Use Python 3.11+
* IDE: VSCode
* All dependencies listed in requirements.txt
* Project structure follows MVC with strategy separation

**Project Functionality**

**User Modes**

* Single-user desktop application
* No network or server dependency

**Main Mechanics / Sub Mechanics**

* Capture real-time audio and process it into spectrum data
* Analyze spectrum/volume to classify current mood
* Update visuals based on classified mood
* Mechanic logic separated using Strategy Pattern

**User Interface / Inputs / Sensors**

* Inputs: Microphone only
* Interface: Real-time mood display (visual background + label)
* Interaction: Dropdown to select mood strategy

**Graphics / Animation / Video**

* Rendered via pygame
* Mood visuals: dynamic color changes with ripple blended with fade effects
* Simple geometric visuals based on mood class

**Audio**

* Real-time capture via sounddevice
* Analysis in mono or stereo buffer
* No prerecorded audio used
* Possible future addition: voice feedback ("Mood is calm")

**Networking**

No network features used in this version.

**Databases / Server-Side Technology**

Not required. Optional: export mood logs to local .csv or .json.

**Localization Plan**

Not applicable for this version. Minimal text shown on UI.

Future: multilingual mood labels.

**Custom Topics**

* Strategy Pattern integration for mood detection logic
* FFT visualization and mapping to mood classification in real time
* Dynamic UI update using audio-derived data

**Additional Technical Information**

* Frame update cycle every 100–200ms
* Audio window buffer: 1024 samples

**Mockups / Flowcharts**

**UI Overview**

* **Top:** App title + current strategy label
* **Center:** Animated background based on mood (color + motion)
* **Bottom:** Current mood label ("Calm", "Energetic", etc.)

**Flowchart:**

Start App

↓

Initialize UI and Audio Input

↓

While Running:

- Capture audio buffer

- Run selected strategy

- Get mood

- Update UI

↓

User closes app → Exit

**Production Documentation**

**Feature List**

| **Number** | **Feature** | **Category** | **Priority** | **Difficulty** |
| --- | --- | --- | --- | --- |
| F1 | Real-time audio input | System | High | Medium |
| F2 | Frequency spectrum visualizer | UI | High | Medium |
| F3 | Mood detection engine | System | High | Hard |
| F4 | Strategy-based detection | System | Medium | Medium |
| F5 | Mood-based visual feedback | UI | High | Medium |

**Master Task List**

| **Number** | **Task Description** | **Feature** | **Dependencies** |
| --- | --- | --- | --- |
| T1 | Capture audio input and test raw data | F1 | None |
| T2 | Implement FFT and display live spectrum | F2 | T1 |
| T3 | Build MoodStrategy interface | F3 | T2 |
| T4 | Implement Volume and Frequency strategies | F4 | T3 |
| T5 | Integrate mood output into visual display | F5 | T4 |
| T6 | Final UI polish and strategy selector | F5 | T5 |

**Development Roadmap**

**Milestone 1: Audio + Visualizer Setup**

* T1: Build microphone input loop using sounddevice
* T2: Capture audio into buffer, convert to numpy array
* T2: Display real-time bar graph using FFT results via pygame
* T2: Adjust update speed and resolution

**Milestone 2: Mood Detection Engine**

* T3: Create base MoodStrategy interface
* T4: Implement VolumeStrategy (based on RMS energy)
* T4: Implement FrequencyStrategy (based on spectral centroid)
* T4: Connect strategy output to mood label display

**Milestone 3: Visual Feedback + Polish**

* T5: Change background visuals based on detected mood
* T5: Smooth transitions between color states
* T6: Add dropdown or hotkey to switch strategies live
* T6: Write 3–5 unit tests for strategy outputs and UI sync
* T6: Prepare final UI for submission

**Verification and Validation (V&V)**

| **Task** | **Verification (Does it work?)** | **Validation (Is it useful?)** |
| --- | --- | --- |
| T1 | Audio data captured as numpy array | Audio reacts to sound changes in the room |
| T2 | FFT processed correctly into frequency bins | Spectrum display updates in real time |
| T3 | MoodStrategy has correct interface | Future strategies can plug in without errors |
| T4 | Mood is returned as string value | Mood feels accurate based on sound |
| T5 | Background color changes with mood | User can tell the room is "Calm" vs "Tense" visually |
| T6 | Strategies can switch mid-session | User can compare results live and choose best |

**References**

1. E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1994.
2. Refactoring.Guru. (n.d.). Strategy Design Pattern. https://refactoring.guru/design-patterns/strategy
3. Full FFT audio processing example in Python: https://docs.scipy.org/doc/scipy/reference/fft.html
4. sounddevice official documentation: <https://python-sounddevice.readthedocs.io>