VaaniMaatu - Step-by-Step Implementation Guide

Ms. Rai, Ms. Lakshmi, and Third Classmate
Sahyadri Engineering College, Mangalore

August 22, 2025, 11:53 AM IST

Introduction

This guide outlines the step-by-step process for implementing VaaniMaatu, an open-source speech therapy application for Kannada speakers. Aimed at developers and contributors, it covers environment setup, audio processing, content integration, and deployment, ensuring replicability and extensibility.

1 Setting Up the Development Environment

- Installing tools for dual modalities.
- Configuring version control for collaboration.

1.1 Android Environment

- Download and install Android Studio (2022.3.1) from developer.android.com.
- Install Kotlin plugin and SDKs (API 21+).
- Add dependencies in ‘build.gradle‘:

```
            dependencies {
                implementation 'androidx.room:room-runtime
                   :2.5.0'
                implementation 'com.github.hiteshsondhi88:
                   ffmpeg-kit-full:4.5.1'
                implementation 'com.github.PhilJay:
                   MPAndroidChart:v3.1.0'
            }
```

- Sync project and verify setup.

1.2 Web Environment

- Install Visual Studio Code (1.81.0) from code.visualstudio.com.
- Add Live Server extension for testing.

- Initialize project directory: ‘mkdir vaanimaatu-mvp; cd vaanimaatu-mvp‘.
- Install Node.js (v18+) for future backend: nodejs.org.

1.3 Version Control

- Initialize Git: ‘git init‘.
- Link to GitHub: ‘git remote add origin https://github.com/sahyadri-team/vaanimaatu‘.
- Commit initial files: ‘git add .; git commit -m "Initial setup"‘.

2 Implementing Audio Processing

- Creating DAF functionality for both platforms.

2.1 Android Audio

- Create ‘FeedbackManager.kt‘ for audio handling.
- Initialize ‘AudioRecord‘ and ‘AudioTrack‘ with 44100 Hz sample rate.
- Implement delay using ‘LinkedBlockingQueue‘:

```
            val delayFrames = (delayMs * 44100 / 1000) / bufferSize
            repeat(delayFrames) { bufferQueue.poll() }
```

- Test with ‘startFeedback()‘ and ‘stopFeedback()‘ methods.

2.2 Web Audio

- Define ‘DAFProcessor‘ class in ‘daf-processor.js‘.
- Set up ‘AudioContext‘ with constraints:

```
            new (window.AudioContext)({ latencyHint: 'interactive', sampleRate: 44100 })
```

- Connect nodes: ‘microphone analyserNode delayNode gainNode destination‘.
- Adjust delay: ‘delayNode.delayTime.value = 0.150‘.

3 Integrating Content

- Adding culturally relevant materials.

3.1 Reading Passages

- Source passages from Kannada educational materials, vetted for phonetic suitability.
- Organize into JSON arrays in ‘assets/content.json‘ (Android) or ‘js/content.js‘ (Web):

```
[
    { "id": 1, "level": "beginner", "text": "    :
        ..." },
    { "id": 2, "level": "intermediate", "text": "
      :     ..." },
    { "id": 3, "level": "advanced", "text": " :
      ..." }
]
```

- Load dynamically into UI dropdowns.

3.2 Images for Naming Tasks

- Curate 98 images from royalty-free sources (e.g., Unsplash) and local contributions.
- Tag with JSON in ‘assets/content.json‘:

```
[
    { "id": 1, "type": "naming", "file": "
      mangalore_beach.jpg", "prompt": "   " }
]
```

- Optimize to <100KB using TinyPNG and store in ‘assets/‘.

3.3 Conversation Prompts

- Develop prompts based on cultural contexts (e.g., "   ").
- Store in JSON and link to UI for real-time feedback.

4 Building the User Interface

- Designing responsive and accessible UI.
- Use Material Design principles for Android layouts (e.g., ‘activity$_{m}ain.xml$‘)$.ImplementCSSforu$

5 Adding Innovative Features

- `– Implementing analytics and gamification.`

- Track session time and fluency metrics using ‘Vosk‘ (Android) or ‘analyserNode‘ (Web).
- Award points (50/session) and badges in ‘app.js‘:

```
            if (this.sessionTime % 15 === 0) {
                this.points += 50;
                if (this.sessionTime === this.dailyGoal *
                   60) this.achievements++;
            }
```

- Store progress in ‘SharedPreferences‘ (Android) or ‘localStorage‘ (Web).

6 Testing and Debugging

- Validating functionality and performance.
- Run unit tests with JUnit (Android) and Jest (Web).
- Conduct integration tests for audio and UI components.
- Optimize latency (<50ms) by adjusting buffer sizes.

7 Deployment

- Releasing the application.
- Build Android APK: ‘Build > Build Bundle(s) / APK(s)‘ in Android Studio.
- Deploy web to GitHub Pages: Push to ‘gh-pages‘ branch.
- Test post-deployment on multiple devices.

8 Conclusion

This guide provides a comprehensive framework for implementing VaaniMaatu. Contributors are encouraged to follow these steps, extending features like FAF/MAF or multilingual support.