Recitation Companion App

A Flutter-based mobile application for learning and practicing Sanskrit verse recitation with authentic pronunciation guidance powered by AI.

🎯 Features

- **Authentic Sanskrit TTS**: Uses Vakyansh model trained specifically on Sanskrit corpus
- **Interactive Verse Learning**: Browse and learn verses from Vedic scriptures
- **Real-time Playback**: High-quality audio synthesis (50-150ms generation time)
- **Progress Tracking**: Monitor your learning journey
- **Beautiful UI**: Saffron-orange themed design reflecting traditional aesthetics
- **Offline Capable**: Works without internet after initial setup

🏗 Architecture

Backend (FastAPI + Python)

- **Framework**: FastAPI
- **TTS Engine**: Coqui TTS with Vakyansh Sanskrit model
- **API**: RESTful endpoints for verse data and TTS generation
- **Port**: 8000

Frontend (Flutter + Dart)

- **Framework**: Flutter
- **State Management**: Riverpod
- **Audio Playback**: just_audio package
- **HTTP Client**: http package

📋 Prerequisites

- **Python 3.8+** (for backend)
- **Flutter 3.0+** (for frontend)
- **2GB disk space** (for TTS model)
- **Windows/Linux/macOS**

🚀 Quick Start

1. Backend Setup (5 minutes)

```
# Navigate to backend directory
cd BACKEND

# Install TTS dependencies (one-time setup)
.\install_tts.bat
```

```
# OR manually:
pip install torch torchvision torchaudio --index-url
https://download.pytorch.org/whl/cpu
pip install TTS fastapi uvicorn httpx pydantic pydantic-settings python-dotenv

# Start the backend server
python -m uvicorn app.main:app --reload --host 0.0.0.0 --port 8000
```

First Request Note: The TTS model (~100MB) downloads automatically on first use. Subsequent requests are instant.

2. Frontend Setup (2 minutes)

```
# Navigate to frontend directory
cd FRONTEND

# Get dependencies
flutter pub get

# Run the app
flutter run
```

🧪 Testing

Test Backend TTS

```
# Health check
Invoke-WebRequest -Uri "http://localhost:8000/api/v1/tts/health"

# Generate Sanskrit audio
Invoke-WebRequest -Uri "http://localhost:8000/api/v1/tts/generate?text=ॐ नमः
शिवाय" -OutFile test.wav

# Play test.wav to hear authentic Sanskrit pronunciation
```

Test API Documentation

Open in browser: http://localhost:8000/docs

📁 Project Structure

```
.
├── BACKEND/
│   ├── app/
│   │   ├── routers/
│   │   │   ├── tts.py          # TTS endpoint (Vakyansh model)
│   │   │   └── verses.py       # Verse data endpoints
│   │   ├── services/
│   │   │   └── vedic_service.py # Vedic scriptures integration
│   │   ├── models/
│   │   │   └── schemas.py      # Pydantic models
│   │   ├── config.py           # Configuration
│   │   └── main.py             # FastAPI application
│   ├── requirements.txt        # Python dependencies
│   ├── install_tts.bat         # TTS installation script
│   └── .env.example            # Environment variables template
│
├── FRONTEND/
│   ├── lib/
│   │   ├── screens/
│   │   │   ├── verse_detail_screen.dart  # Main playback UI
│   │   │   ├── home_screen.dart          # Dashboard
│   │   │   └── ...                       # Other screens
│   │   ├── services/
│   │   │   ├── tts_service.dart          # TTS API wrapper
│   │   │   ├── api_service.dart          # HTTP client
│   │   │   └── api_config.dart           # API configuration
│   │   ├── providers/
│   │   │   └── ...                       # Riverpod providers
│   │   └── models/
│   │       └── ...                       # Data models
│   └── pubspec.yaml            # Flutter dependencies
│
└── README.md                   # This file
```

🔧 Configuration

Backend Configuration

Edit `BACKEND/.env`:

```
# Server settings
HOST=0.0.0.0
PORT=8000
RELOAD=true

# API settings
VEDIC_API_BASE_URL=https://vedicscriptures.github.io
CORS_ORIGINS=http://localhost:3000,http://localhost:8080
```

Frontend Configuration

Edit `FRONTEND/lib/services/api_config.dart`:

```
class ApiConfig {
  static const String baseUrl = 'http://localhost:8000';
  // Change to your backend URL if different
}
```

📊 Performance

Metric	Value
TTS Model Loading	~2-3 seconds (first request)
Audio Generation	50-150ms
Total API Response	200-300ms
Model Size	~100MB
Supported Languages	Sanskrit (primary), Hindi, English

🛠 Troubleshooting

Backend Issues

"Import TTS.api could not be resolved"

```
pip install TTS
```

"RuntimeError: Couldn't load custom C++ ops"

- This warning is normal for CPU inference
- Audio generation will still work

Slow first request

- Model downloads automatically (~100MB)
- Subsequent requests are fast

Frontend Issues

"Connection refused"

- Ensure backend is running on port 8000
- Check `api_config.dart` has correct URL

Audio doesn't play

- Check backend logs for errors
- Test endpoint directly: `/api/v1/tts/generate?text=test`

- Verify just_audio package is installed

General

Out of memory

- Close unnecessary applications
- Reduce concurrent TTS requests
- Consider using smaller batch sizes

🔑 API Endpoints

TTS Endpoints

Generate Speech (POST)

```
POST /api/v1/tts/generate
Content-Type: application/json

{
  "text": "ॐ नमः शिवाय"
}

Response: audio/wav
```

Generate Speech (GET)

```
GET /api/v1/tts/generate?text=ॐ नमः शिवाय

Response: audio/wav
```

Health Check

```
GET /api/v1/tts/health

Response:
{
  "status": "healthy",
  "model": "Vakyansh Sanskrit TTS (FastPitch)",
  "provider": "Coqui TTS",
  "loaded": true,
  "ready": true
}
```

Verse Endpoints

Get All Verses

```
GET /api/v1/verses

Response: Array of verse objects
```

Get Verse by ID

```
GET /api/v1/verses/{verse_id}

Response: Verse object with audio data
```

🎨 Design System

The app uses a traditional saffron-orange color scheme:

- **Primary**: #FF6B35 (Saffron Orange)
- **Accent**: #F7931E (Golden Orange)
- **Deep**: #D84315 (Deep Orange)
- **Light**: #FFAB91 (Light Saffron)

📱 Supported Platforms

- ✅ Android
- ✅ iOS
- ✅ Windows
- ✅ macOS
- ✅ Linux
- ✅ Web

🤝 Contributing

1. Fork the repository
2. Create feature branch (`git checkout -b feature/AmazingFeature`)
3. Commit changes (`git commit -m 'Add AmazingFeature'`)
4. Push to branch (`git push origin feature/AmazingFeature`)
5. Open Pull Request

📝 License

This project is licensed under the MIT License.

🙏 Acknowledgments

- **Vakyansh**: Sanskrit TTS model training

- **Coqui AI**: TTS engine and infrastructure
- **Vedic Scriptures API**: Verse data source
- **Flutter Team**: Cross-platform framework

📧 Support

For issues and questions:

1. Check this README thoroughly
2. Review backend logs: `BACKEND/` terminal output
3. Check Flutter console for errors
4. Test endpoints using `/docs` interface

🔮 Future Enhancements

- [ ] Real-time pronunciation feedback using ASR
- [ ] Verse memorization challenges
- [ ] Community features and leaderboards
- [ ] Offline verse library
- [ ] Custom practice routines
- [ ] Advanced pronunciation analysis

Made with 💖 for Sanskrit learners worldwide