

# Vedam Tutorials - Lesson Management Guide

## Table of Contents

1. [Project Structure](#)
  2. [JSON Configuration System](#)
  3. [Adding New Lessons](#)
  4. [File Naming Conventions](#)
  5. [Image Requirements](#)
  6. [Audio Requirements](#)
  7. [Deployment](#)
  8. [Troubleshooting](#)
- 

## Project Structure

Your Next.js project follows this folder structure with JSON configuration files:



```

vedam-tutorials/
├── public/
│   └── lessons/
│       ├── index.json          # Main lessons list
│       └── arunam/
│           ├── config.json     # Arunam lesson configuration
│           ├── anuvakam-1/
│           │   ├── image.jpg
│           │   ├── audio.mp3
│           │   └── panasa-1/
│           │       ├── image.jpg
│           │       └── audio.mp3
│           │   └── panasa-2/
│           │       ├── image.jpg
│           │       └── audio.mp3
│           └── anuvakam-2/
│               ├── image.jpg
│               ├── audio.mp3
│               └── panasa-1/
│                   ├── image.jpg
│                   └── audio.mp3
│       └── rudram/
│           ├── config.json     # Rudram lesson configuration
│           └── [anuvakams and panasas...]
├── src/
│   └── app/
│       ├── page.js (main component)
│       └── layout.js
├── package.json
└── next.config.js

```

# JSON Configuration System

## Main Lessons Index (/public/lessons/index.json)

This file lists all available main lessons. The application reads this file on startup to display the home page.

**Location:** /public/lessons/index.json

**Format:**



json

```
[
  {
    "id": "arunam",
    "title": "Arunam",
    "description": "First main lesson of Vedam"
  },
  {
    "id": "rudram",
    "title": "Rudram",
    "description": "The powerful hymn to Lord Shiva"
  },
  {
    "id": "chamakam",
    "title": "Chamakam",
    "description": "The prayer of desires"
  }
]
```

Field Descriptions:

- id: Unique identifier (lowercase, no spaces, used in folder names)
- title: Display name shown to users
- description: Brief description of the lesson

Lesson Configuration (/public/lessons/[lesson-id]/config.json)

Each main lesson has its own config.json file. **The structure is flexible** - lessons can have:

- Both anuvakams and panasas (full structure)
- Only anuvakams (no panasas breakdown)
- Neither anuvakams nor panasas (direct lesson)

Option 1: Full Structure (Anuvakams with Panasas)

**Location:** /public/lessons/arunam/config.json



json

```
{
  "id": "arunam",
  "title": "Arunam",
  "description": "First main lesson of Vedam",
  "anuvakams": [
    {
      "id": "anuvakam-1",
      "title": "Anuvakam 1",
      "panasas": [
        {
          "id": "panasa-1",
          "title": "Panasa 1"
        },
        {
          "id": "panasa-2",
          "title": "Panasa 2"
        }
      ]
    }
  ]
}
```

**Option 2: Anuvakams Only (No Panasas)**

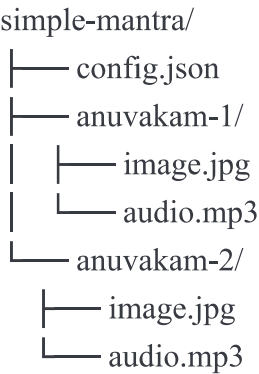
**Location:** /public/lessons/simple-mantra/config.json



json

```
{
  "id": "simple-mantra",
  "title": "Simple Mantra",
  "description": "A mantra with sections but no subdivisions",
  "anuvakams": [
    {
      "id": "anuvakam-1",
      "title": "First Section"
    },
    {
      "id": "anuvakam-2",
      "title": "Second Section"
    }
  ]
}
```

Folder Structure:



Option 3: Direct Lesson (No Anuvakams or Panasas)

Location: /public/lessons/short-prayer/config.json



json

```
{
  "id": "short-prayer",
  "title": "Short Prayer",
  "description": "A simple prayer without subdivisions"
}
```

Folder Structure:



```
short-prayer/
├── config.json
├── image.jpg
└── audio.mp3
```

**Note:** The anuvakams field is optional. If omitted or empty, the lesson displays as a single unit.

---

# Adding New Lessons

## Step-by-Step Process

### Step 1: Update Main Index File

- 1. Open /public/lessons/index.json
- 2. Add your new lesson to the array:



json

```
[
  {
    "id": "arunam",
    "title": "Arunam",
    "description": "First main lesson of Vedam"
  },
  {
    "id": "your-new-lesson",
    "title": "Your New Lesson",
    "description": "Description of your new lesson"
  }
]
```

**Step 2: Create Lesson Folder Structure**



bash

```
cd public/lessons
mkdir your-new-lesson
cd your-new-lesson
```

**Step 3: Create Lesson Configuration File**

Create /public/lessons/your-new-lesson/config.json:



json

```
{
  "id": "your-new-lesson",
  "title": "Your New Lesson",
  "description": "Description of your new lesson",
  "anuvakams": [
    {
      "id": "anuvakam-1",
      "title": "Anuvakam 1",
      "panasas": [
        {
          "id": "panasa-1",
          "title": "Panasa 1"
        },
        {
          "id": "panasa-2",
          "title": "Panasa 2"
        }
      ]
    }
  ]
}
```

**Step 4: Create Folder Structure for Media Files**



bash

```
# Create anuvakam folders
mkdir anuvakam-1
mkdir anuvakam-2

# Create panasa folders inside each anuvakam
mkdir anuvakam-1/panasa-1
mkdir anuvakam-1/panasa-2
mkdir anuvakam-2/panasa-1
```

**Step 5: Add Media Files**

For each anuvakam, add:

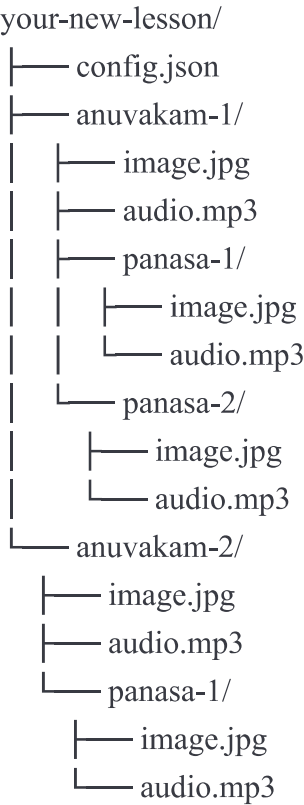


- image.jpg - Full anuvakam text as image
- audio.mp3 - Complete anuvakam recitation

For each panasa, add:

- image.jpg - Panasa text as image
- audio.mp3 - Panasa recitation

**Example final structure:**



# File Naming Conventions

## Folder Names

- **Main Lessons:** Use lowercase with hyphens
  - arunam, rudram, sri-rudram
  - Arunam, RUDRAM, sri\_rudram
- **Anuvakams:** Use anuvakam- prefix with number
  - anuvakam-1, anuvakam-2, anuvakam-10
  - Anuvakam1, anuvakam\_1, anuv-1
- **Panasas:** Use panasa- prefix with number
  - panasa-1, panasa-2, panasa-15
  - Panasa1, panasa\_1, pan-1

## File Names

- **Images:** Always image.jpg or image.png
- **Audio:** Always audio.mp3
- **Config:** Always config.json for lessons, index.json for main list

## JSON IDs

- Use lowercase with hyphens
  - Must match folder names exactly
  - Examples: "arunam", "anuvakam-1", "panasa-3"
- 

## Image Requirements

### Format

- **Preferred:** JPG or PNG
- **Resolution:** Minimum 1200px width for clarity
- **Aspect Ratio:** 16:9 or 4:3 recommended
- **File Size:** Keep under 2MB for faster loading

### Content Guidelines

- Use clear, high-contrast text
- Ensure Sanskrit/Devanagari script is legible
- Include proper diacritical marks
- Use consistent font size across all lessons
- Consider both desktop and mobile viewing

### Recommended Tools

- **Creating Images:** Canva, Adobe Photoshop, GIMP
  - **Scanning Text:** Use 300 DPI minimum
  - **Optimization:** TinyPNG, ImageOptim, Squoosh
- 

## Audio Requirements

### Format

- **Format:** MP3
- **Bitrate:** 128 kbps minimum (192 kbps recommended)
- **Sample Rate:** 44.1 kHz
- **Channels:** Mono or Stereo

### Recording Guidelines

- Record in a quiet environment
- Use consistent volume levels across all recordings
- Remove background noise
- Add 1-2 seconds of silence at start and end

- Keep clear pronunciation with appropriate pace

## Recommended Tools

- **Recording:** Audacity (free), Adobe Audition
- **Editing:** Audacity, GarageBand
- **Conversion:** FFmpeg, Online-Convert.com

## Audio Processing Commands



bash

```
# Convert WAV to MP3 using FFmpeg
ffmpeg -i input.wav -codec:a libmp3lame -b:a 192k output.mp3

# Normalize audio levels
ffmpeg -i input.mp3 -af "loudnorm" output.mp3

# Batch convert all WAV files in folder
for file in *.wav; do
    ffmpeg -i "$file" -codec:a libmp3lame -b:a 192k "${file%.wav}.mp3"
done
```

---

## Complete Example: Adding "Rudram" Lesson

### 1. Update /public/lessons/index.json



json

```
[
  {
    "id": "arunam",
    "title": "Arunam",
    "description": "First main lesson of Vedam"
  },
  {
    "id": "rudram",
    "title": "Rudram",
    "description": "The powerful hymn to Lord Shiva"
  }
]
```

2. Create /public/lessons/rudram/config.json

Example with Full Structure:



json

```
{
  "id": "rudram",
  "title": "Rudram",
  "description": "The powerful hymn to Lord Shiva",
  "anuvakams": [
    {
      "id": "anuvakam-1",
      "title": "Anuvakam 1",
      "panasas": [
        {
          "id": "panasa-1",
          "title": "Panasa 1"
        },
        {
          "id": "panasa-2",
          "title": "Panasa 2"
        }
      ]
    }
  ],
  {
    "id": "anuvakam-2",
    "title": "Anuvakam 2",
    "panasas": [
      {
        "id": "panasa-1",
        "title": "Panasa 1"
      }
    ]
  }
]
```

### 3. Create Folder Structure

For Full Structure (with panasas):



bash

```
mkdir -p public/lessons/rudram/anuvakam-1/panasa-{1,2}
```

```
mkdir -p public/lessons/rudram/anuvakam-2/panasa-1
```

**For Anuvakams Only** (no panasas):



bash

```
mkdir -p public/lessons/rudram/anuvakam-{1,2}
```

**For Direct Lesson** (no subdivisions):



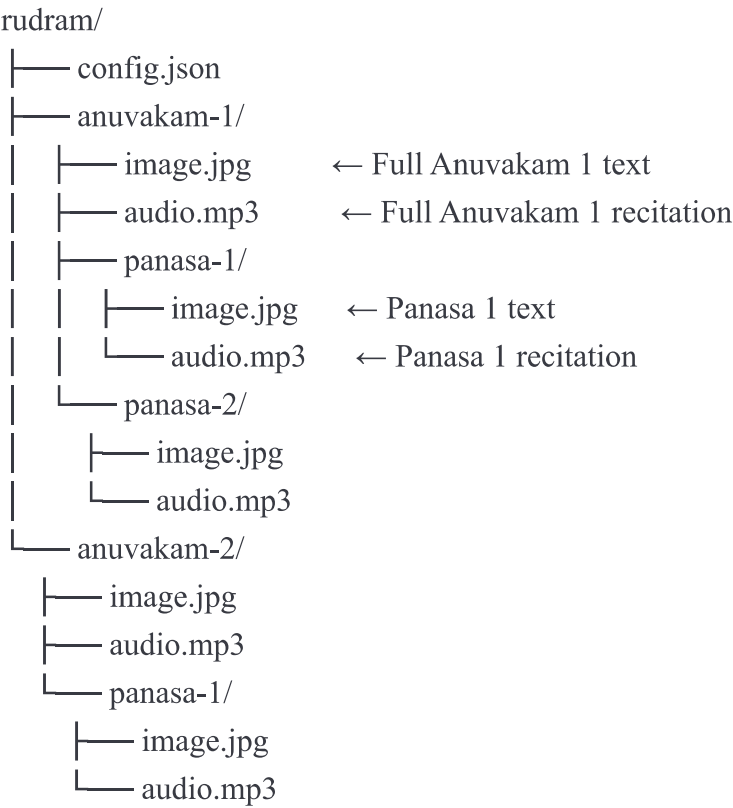
bash

```
mkdir public/lessons/rudram
```

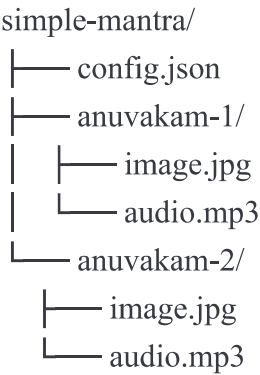
**4. Add Media Files**

**Full Structure Example:**





**Anuvakams Only Example:**



**Direct Lesson Example:**



```
short-prayer/  
├── config.json  
├── image.jpg  
└── audio.mp3
```

---

## File Path Examples

The application automatically generates file paths based on the JSON configuration:

### Full Structure (with Anuvakams and Panasas)

- **Anuvakam Image:** /lessons/{lesson-id}/{anuvakam-id}/image.jpg
- **Anuvakam Audio:** /lessons/{lesson-id}/{anuvakam-id}/audio.mp3
- **Panasa Image:** /lessons/{lesson-id}/{anuvakam-id}/{panasa-id}/image.jpg
- **Panasa Audio:** /lessons/{lesson-id}/{anuvakam-id}/{panasa-id}/audio.mp3
- **Example:** /lessons/rudram/anuvakam-1/panasa-2/audio.mp3

### Anuvakams Only (no Panasas)

- **Anuvakam Image:** /lessons/{lesson-id}/{anuvakam-id}/image.jpg
- **Anuvakam Audio:** /lessons/{lesson-id}/{anuvakam-id}/audio.mp3
- **Example:** /lessons/simple-mantra/anuvakam-1/image.jpg

### Direct Lesson (no Anuvakams or Panasas)

- **Lesson Image:** /lessons/{lesson-id}/image.jpg
  - **Lesson Audio:** /lessons/{lesson-id}/audio.mp3
  - **Example:** /lessons/short-prayer/audio.mp3
- 

## Deployment

### Local Development

1. **Install dependencies:**



bash

`npm install`

2. **Run development server:**



bash



`npm run dev`

3. **Access application:** `http://localhost:3000`

## Production Build

1. **Create optimized build:**



bash

`npm run build`

2. **Test production build:**



bash

`npm start`

## Deployment Platforms

### Vercel (Recommended)



bash

*# Install Vercel CLI*

`npm install -g vercel`

*# Deploy*

`vercel`

### Netlify

- Connect GitHub repository
- Build command: `npm run build`
- Publish directory: `.next`

### Self-Hosted with PM2



bash

```
npm run build
pm2 start npm --name "vedam-tutorials" -- start
```

# Troubleshooting

## JSON Configuration Issues

**Problem: Lesson not appearing on home page**

**Solutions:**

- Check /public/lessons/index.json is valid JSON
- Verify lesson ID in index matches folder name exactly
- Restart development server after changes
- Check browser console for errors

**Problem: Anuvakams or Panasas not loading**

**Solutions:**

- Verify config.json exists in lesson folder
- Check JSON syntax is valid (use JSONLint.com)
- Ensure IDs match folder names exactly (case-sensitive)
- Check for trailing commas in JSON (not allowed)

## Media File Issues

**Problem: Images not loading**

**Solutions:**

- Verify file is named exactly image.jpg or image.png
- Check file is in correct folder based on config.json
- Ensure file path is case-sensitive correct
- Clear browser cache

**Problem: Audio not playing**

**Solutions:**

- Verify file is named exactly audio.mp3
- Check audio format is valid MP3
- Test audio file in media player first
- Check browser console for 404 errors

# Common JSON Errors

## Missing Comma



json

```
// ❌ Wrong
{
  "id": "arunam"
  "title": "Arunam"
}
```

```
// ✅ Correct
{
  "id": "arunam",
  "title": "Arunam"
}
```

## Trailing Comma



json

```
// ❌ Wrong
{
  "id": "arunam",
  "title": "Arunam",
}
```

```
// ✅ Correct
{
  "id": "arunam",
  "title": "Arunam"
}
```

## Wrong Quotes



json

```
// ❌ Wrong (single quotes)
{
  'id': 'arunam'
}
```

```
// ✅ Correct (double quotes)
{
  "id": "arunam"
}
```

## Validation Script

Create a script to validate your JSON configuration:



javascript

```
// validate-lessons.js

const fs = require('fs');
const path = require('path');

const lessonsDir = './public/lessons';

// Read main index
const indexPath = path.join(lessonsDir, 'index.json');
const mainLessons = JSON.parse(fs.readFileSync(indexPath, 'utf8'));

console.log(`Found ${mainLessons.length} main lessons`);

mainLessons.forEach(lesson => {
  const configPath = path.join(lessonsDir, lesson.id, 'config.json');

  if (!fs.existsSync(configPath)) {
    console.error(` ❌ Missing config.json for lesson: ${lesson.id}`);
    return;
  }

  const config = JSON.parse(fs.readFileSync(configPath, 'utf8'));
  console.log(` ✅ ${lesson.title}: ${config.anuvakams.length} anuvakams`);

  config.anuvakams.forEach(anuvakam => {
    console.log(` - ${anuvakam.title}: ${anuvakam.panasas.length} panasas`);
  });
});
```

Run with: node validate-lessons.js

# Quick Reference Checklist

When adding a new lesson:

- ☐ Updated /public/lessons/index.json
- ☐ Created lesson folder: /public/lessons/[lesson-id]/
- ☐ Created config.json in lesson folder
- ☐ Validated JSON syntax (no errors)
- ☐ All IDs match folder names exactly
- ☐ Created all anuvakam folders
- ☐ Created all panasa folders
- ☐ Each anuvakam has image.jpg and audio.mp3

- ☐ Each panasa has image.jpg and audio.mp3
- ☐ Images are clear and properly sized
- ☐ Audio files play correctly
- ☐ Tested on development server
- ☐ Tested on mobile device
- ☐ Deployed to production
- ☐ Verified on production site

## Tips for Efficiency

### Batch Creating Folders



bash

```
# Create multiple anuvakams at once
for i in {1..11}; do mkdir -p "anuvakam-$i"; done

# Create multiple panasas inside anuvakam-1
for i in {1..5}; do mkdir -p "anuvakam-1/panasa-$i"; done
```

### JSON Template Generator



bash

*# Create a template config.json with 5 anuvakams*

```
cat > config.json << 'EOF'
{
  "id": "new-lesson",
  "title": "New Lesson",
  "description": "Description here",
  "anuvakams": [
    {
      "id": "anuvakam-1",
      "title": "Anuvakam 1",
      "panasas": [
        {"id": "panasa-1", "title": "Panasa 1"},
        {"id": "panasa-2", "title": "Panasa 2"}
      ]
    }
  ]
}
EOF
```

---

## Support and Maintenance

### Regular Tasks

- Backup JSON files and media weekly
- Validate JSON configuration monthly
- Update audio recordings if needed
- Monitor user feedback
- Check for broken links/files

### Version Control Best Practices



bash

*# Always commit JSON changes*

```
git add public/lessons/index.json
```

```
git add public/lessons/*/config.json
```

```
git commit -m "Add new lesson: Rudram"
```

*# Tag releases*

```
git tag -a v1.1.0 -m "Added Rudram lesson"
```

```
git push --tags
```

---

*Last Updated: November 2025*