

TTS Campaign Episode Processor - Detailed Implementation Outline

0_0_0 Project Foundation

Overview

Convert D&D campaign scripts to individual character audio files using ElevenLabs TTS, with smart parsing, cost tracking, and professional audio output for mixing.

Core Requirements

- Parse both `**NAME:**` and `(NAME:)` speaker formats
- Remove italic stage directions/actions
- Generate individual numbered audio files per character
- Track ElevenLabs subscription quota vs overage costs
- No hard spending limits, just clear warnings

Development Guidelines

- **CRITICAL:** Update README.md after EVERY checkpoint completion
 - README must track: Current phase, completed checkpoints, next steps
 - Use checkbox format: `- [x] 1_1_1 Initialize Project`
 - Include timestamp for each completion
 - Document any deviations or issues encountered
-

PHASE 1: Core Infrastructure

1_1_0 Project Setup & Structure

1_1_1 Initialize Project

bash

```
campaign-tts-processor/
├─ frontend/
│   ├─ index.html
│   ├─ css/
│   │   └─ styles.css
│   └─ js/
│       ├─ app.js
│       ├─ parser.js
│       ├─ speaker-mapper.js
│       └─ processor.js
├─ backend/
│   ├─ server.js
│   ├─ routes/
│   │   ├─ upload.js
│   │   ├─ process.js
│   │   └─ elevenlabs.js
│   └─ utils/
│       ├─ script-parser.js
│       ├─ cost-calculator.js
│       └─ file-manager.js
├─ outputs/
│   └─ [episode_folders]/
├─ .env.example
├─ package.json
└─ README.md
```

1_1_2 Dependencies & Environment

- **Frontend:** Vanilla JS, CSS Grid/Flexbox
- **Backend:** Express, Multer, dotenv
- **API:** ElevenLabs SDK
- **Dev:** Nodemon, cors
- **Environment Variables:**

```
ELEVENLABS_API_KEY=  
ELEVENLABS_SUBSCRIPTION_QUOTA=1000000  
PORT=3000
```

1_1_3 Basic Server Setup

- Express server with CORS
- Static file serving

- Environment variable loading
- Basic error handling middleware
- Health check endpoint

1_2_0 File Upload System

1_2_1 Frontend Upload Interface

```
html

<!-- Clean drag-and-drop zone -->
<div class="upload-zone">
  <input type="file" accept=".md,.txt,.doc,.docx">
  <p>Drop your episode script here</p>
  <p class="supported">Supports: MD, TXT, DOC, DOCX</p>
</div>
```

1_2_2 File Processing Pipeline

- Multer configuration for file uploads
- File type validation
- Convert DOC/DOCX to plain text
- Store in temp directory
- Return file ID for processing

1_2_3 Initial File Preview

- Display first 500 characters
- Show file stats (size, estimated characters)
- Detect episode number/name if present
- "Proceed to Parse" button

PHASE 2: Script Parsing Engine

2_1_0 Speaker Detection System

2_1_1 Core Parser Logic

javascript

```
class ScriptParser {
  constructor() {
    // Regex patterns for speaker detection
    this.speakerPatterns = [
      /\^*\^*([A-Z][A-Z\s]+)\^*\^*/gm, // **SPEAKER**:
      /\^([A-Z][A-Z\s]+):/gm          // SPEAKER:
    ];
  }

  detectSpeakers(text) {
    // Smart detection handling both formats
    // Return unique speaker list
  }
}
```

2_1_2 Dialogue Extraction

- Extract all dialogue for each speaker
- Maintain chronological order
- Handle multi-line dialogue correctly
- Count dialogue instances per speaker

2_1_3 Stage Direction Removal

javascript

```
// Remove italic sections but preview what's being removed
cleanDialogue(text) {
  const stageDirections = [];
  const cleaned = text.replace(/\^*([^\^]+)\^*/g, (match, p1) => {
    stageDirections.push(p1);
    return '';
  });
  return { cleaned, removed: stageDirections };
}
```

2_2_0 Parse Preview System

2_2_1 Visual Parse Preview

Parse Preview
Detected 8 speakers:
• JOE (112 lines)
• CELESTIA (89 lines)
• THRENOS (76 lines)
• SAXY (64 lines)
• RAARH (58 lines)
• ALDEN (52 lines)
• MITE (48 lines)
• TRONALD (41 lines)
Stage directions found: 23
[Show removed content]

2_2_2 Removed Content Preview

- Collapsible section showing all removed stage directions
- Red highlighting in context
- Option to manually override removals

2_2_3 Parse Validation

- Check for common parsing errors
- Warn about unusually short/long dialogues
- Flag potential speaker name variations
- "Confirm Parse" / "Adjust Settings" options

PHASE 3: Speaker-to-Voice Mapping

















3_1_0 Voice Selection Interface

3_1_1 ElevenLabs Voice Fetching

```
javascript
```

```
async fetchAvailableVoices() {
  // Get all voices from ElevenLabs account
  // Include both default and cloned voices
  // Cache voice list for session
}
```

3_1_2 Speaker Mapping UI

Speaker → Voice Mapping			
JOE (DM)	→ [Daniel	▼]	 
CELESTIA	→ [Rachel	▼]	 
THRENOS	→ [Antoni	▼]	 
SAXY	→ [Bella	▼]	 
RAARH	→ [Adam	▼]	 
ALDEN	→ [Sam	▼]	 
MITE	→ [Josh	▼]	 
TRONALD	→ [Clyde	▼]	 
[Save Mapping] [Load Previous]			

3_1_3 Voice Preview System

- Test each voice with sample dialogue from that character
- Play button generates 1-2 sentence preview
- Settings gear for voice parameters (stability, similarity)

3_2_0 Voice Configuration

3_2_1 Per-Voice Settings

javascript

```
voiceSettings = {  
  speaker: "JOE",  
  voiceId: "21m00Tcm4TlvDq8ikWAM",  
  parameters: {  
    stability: 0.75,  
    similarity_boost: 0.75,  
    style: 0.5,  
    use_speaker_boost: true  
  }  
}
```

3_2_2 Mapping Persistence

- Save mappings to browser localStorage
- Export/import mapping JSON files
- Quick-load previous episode mappings

- Default mapping suggestions

3_2_3 Validation & Warnings

- Ensure all speakers have assigned voices
 - Warn about duplicate voice assignments
 - Flag if using same voice for multiple characters
 - "Ready to Process" indicator
-

PHASE 4: Cost Calculation & Warnings

4_1_0 Character Counting System

4_1_1 Accurate Character Count

javascript

```
calculateTotalCharacters() {  
  let total = 0;  
  let breakdown = {};  
  
  dialogues.forEach(d => {  
    const chars = d.cleaned.length;  
    total += chars;  
    breakdown[d.speaker] = (breakdown[d.speaker] || 0) + chars;  
  });  
  
  return { total, breakdown };  
}
```



4_1_2 Quota vs Overage Calculator

javascript

```
calculateCost(characters) {
  const quota = process.env.ELEVENLABS_SUBSCRIPTION_QUOTA;
  const used = await getMonthlyUsage();
  const remaining = quota - used;

  if (characters <= remaining) {
    return {
      cost: 0,
      message: `${characters.toLocaleString()} characters ($0.00 - within subscription)`,
      withinQuota: true
    };
  } else {
    const overage = characters - remaining;
    const cost = overage * 0.00003; // $30 per 1M chars
    return {
      cost: cost,
      message: `${characters.toLocaleString()} characters (${cost.toFixed(2)} overage)`,
      withinQuota: false,
      quotaUsed: remaining,
      overageChars: overage
    };
  }
}
```

4_1_3 Cost Preview Display

Processing Cost Estimate
Total Characters: 156,892
Subscription Remaining: 243,108
 Cost: \$0.00 (within subscription)
Character Breakdown:
• JOE: 62,341 (39.7%)
• CELESTIA: 28,921 (18.4%)
• THRENOS: 22,156 (14.1%)
• Others: 43,474 (27.8%)
 Detailed Breakdown

4_2_0 Warning Thresholds

4_2_1 Soft Warning at \$100

javascript

```
if (cost > 100) {  
  showWarning({  
    title: "High Cost Alert",  
    message: `This will cost ${cost.toFixed(2)} in overage charges.`,  
    type: "soft",  
    actions: [  
      { text: "Continue Anyway", class: "warning" },  
      { text: "Review Settings", class: "secondary" },  
      { text: "Cancel", class: "default" }  
    ]  
  });  
}
```

4_2_2 Progressive Warnings

- Green: \$0-25 (within normal range)
- Yellow: \$25-100 (elevated cost)
- Orange: \$100-250 (high cost, soft warning)
- Red: \$250+ (very high cost, strong warning)

4_2_3 Quota Tracking

- Show monthly quota usage graph
- Estimate quota reset date
- Project if current episode will exceed quota
- Historical usage trends

PHASE 5: Processing Engine

5_1_0 Dialogue Processing System

5_1_1 Processing Queue Structure

javascript

```
class ProcessingQueue {
  constructor() {
    this.queue = [];
    this.processed = [];
    this.failed = [];
    this.currentIndex = 0;
  }

  addDialogue(speaker, text, index) {
    this.queue.push({
      id: `${String(index).padStart(3, '0')}_${speaker}`,
      speaker,
      text,
      originalIndex: index,
      status: 'pending'
    });
  }
}
```

5_1_2 ElevenLabs API Integration

javascript

```
async processDialogue(item) {
  try {
    const audio = await elevenlabs.textToSpeech({
      text: item.text,
      voice_id: voiceMapping[item.speaker].voiceId,
      model_id: "eleven_monolingual_v1",
      voice_settings: voiceMapping[item.speaker].parameters
    });

    return {
      ...item,
      audio,
      status: 'completed',
      filename: `${item.id}.mp3`
    };
  } catch (error) {
    return {
      ...item,
      status: 'failed',
      error: error.message
    };
  }
}
```

5_1_3 Chunking for Long Dialogue

javascript

```
chunkLongDialogue(text, maxChars = 1000) {
  if (text.length <= maxChars) return [text];

  const chunks = [];
  const paragraphs = text.split(/\n\n+/);
  let currentChunk = '';

  // Smart paragraph-based chunking
  // Never split mid-sentence
  // Optimal chunk size: 700-1000 chars
}
```

5_2_0 Progress Tracking

5_2_1 Real-time Progress UI

Processing Episode 117A
<div> <div></div> <div></div> </div> 156/312 (50%)
Current: 156_JOE.mp3
Speaker: JOE
Characters: 847
<div> <div>✔</div> <div>Completed: 155</div> </div>
<div> <div>⌚</div> <div>Processing: 1</div> </div>
<div> <div>⏸</div> <div>Queued: 156</div> </div>
<div> <div>✖</div> <div>Failed: 0</div> </div>
Time Elapsed: 3:42
Est. Remaining: 3:38
[Pause] [Cancel]

5_2_2 Error Recovery

javascript

```

async retryFailed() {
  for (const item of this.failed) {
    await delay(1000); // Rate Limit respect
    const result = await processDialogue(item);
    if (result.status === 'completed') {
      this.failed.remove(item);
      this.processed.push(result);
    }
  }
}

```

5_2_3 Process Interruption Handling

- Save progress to localStorage
- Resume capability
- Graceful cancellation
- Partial download option

PHASE 6: File Management & Output

6_1_0 Output Organization






6_1_1 File Naming Convention

```
outputs/  
└─ Episode_117A_2024-11-08/  
   │ 001_JOE.mp3  
   │ 002_CELESTIA.mp3  
   │ 003_THRENOS.mp3  
   │ 004_SAXY.mp3  
   │ 005_RAARH.mp3  
   │ 006_ALDEN.mp3  
   │ 007_MITE.mp3  
   │ 008_TRONALD.mp3  
   │ 009_JOE.mp3  
   │ processing_log.json  
   └─ speaker_mapping.json
```

6_1_2 Metadata Preservation

```
javascript  
  
const metadata = {  
  episode: "117A",  
  processedAt: new Date().toISOString(),  
  totalDialogues: 312,  
  totalCharacters: 156892,  
  cost: 0,  
  speakerMapping: voiceMapping,  
  processingTime: "7:23",  
  failedItems: [],  
  settings: {  
    model: "eleven_monolingual_v1",  
    chunkingEnabled: true,  
    maxChunkSize: 1000  
  }  
};
```

6_1_3 Download Options

 Processing Complete!
Episode 117A
312 files generated
Total time: 7:23
Total cost: \$0.00
Download Options:
• [ Download All (ZIP)]
• [ Download by Speaker]
• [ Download Metadata]
• [ Copy Shareable Link]
Files available for 7 days

6_2_0 Post-Processing Tools

6_2_1 Quality Check Interface

- List any failed conversions
- Preview random samples
- Verify speaker assignments
- Check for silent/corrupt files

6_2_2 Batch Operations

- Re-process failed items
- Bulk download by speaker
- Generate edit decision list (EDL)
- Export for specific DAW formats

6_2_3 Analytics Dashboard

Episode Statistics:

- Average dialogue length: 267 chars
- Longest dialogue: JOE - 2,341 chars
- Speaker balance: [visual chart]
- Processing speed: 42 dialogues/minute
- API response time: 217ms average

PHASE 7: Advanced Features

7_1_0 Smart Enhancements

7_1_1 Context-Aware Chunking

- Detect natural speech patterns
- Preserve dramatic pauses
- Smart paragraph grouping
- Emotion-aware splits

7_1_2 Voice Consistency

- Auto-detect speaker variations (JOE vs JOE (DM))
- Merge similar speaker names
- Suggest voice based on character type
- Remember voice selections across episodes

7_1_3 Preprocessing Options

- Remove dice roll results: (rolls 17)
- Clean OOC comments: (OOO: checking rules)
- Standardize character names
- Fix common OCR errors

7_2_0 Integration Features

7_2_1 DAW Integration

- Export markers for Adobe Audition
- Generate Reaper project files
- Create Pro Tools session data
- Timeline metadata export

7_2_2 Cloud Storage

- Direct upload to Google Drive
- Dropbox integration
- S3 bucket support
- Automatic backup option

7_2_3 Collaboration Tools

- Share processing sessions

- Team voice mapping
 - Approval workflows
 - Version control for scripts
-

PHASE 8: Testing & Deployment

8_1_0 Testing Strategy

8_1_1 Unit Tests

- Parser accuracy tests
- Cost calculation verification
- API mock testing
- File system operations

8_1_2 Integration Tests

- Full episode processing
- Error recovery scenarios
- Rate limit handling
- Concurrent processing

8_1_3 User Acceptance Tests

- Process real episode files
- Verify audio quality
- Test all edge cases
- Performance benchmarks

8_2_0 Deployment

8_2_1 Production Setup

- Environment configuration
- SSL certificates
- Domain setup
- CDN for static assets

8_2_2 Monitoring

- API usage tracking
- Error logging (Sentry)

- Performance metrics
- User analytics







8_2_3 Maintenance

- Automated backups
 - Update procedures
 - Security patches
 - Documentation updates
-

Implementation Priority Order

1. **Phase 1-2:** Foundation (Days 1-2)
 2. **Phase 3-4:** Core Features (Days 3-4)
 3. **Phase 5-6:** Processing Engine (Days 5-7)
 4. **Phase 7:** Enhancements (Week 2)
 5. **Phase 8:** Polish & Deploy (Week 2)
-

Success Metrics

-  Parse accuracy > 99%
-  Processing speed < 30s per dialogue
-  Zero data loss
-  Cost tracking accuracy 100%
-  User can process full episode in < 10 minutes
-  Output files ready for DAW import