Fionntán Setup Guide for Fedora

Complete setup guide to get the AI podcast generator working on Fedora and generate a physics podcast as a test.

Prerequisites

Before starting, ensure you have:

- A Fedora system (tested on Fedora 38+)
- Git installed (`sudo dnf install git`)
- A Google account for API access
- Internet connection

Step 1: Initial Setup (5 minutes)

Clone and Run Setup Script

```bash
# Navigate to your desired directory
cd ~/Desktop  # or wherever you want the project

# The repo should already be cloned, but if not:
# git clone https://github.com/your-username/fionntan.git

cd fionntan

# Run the Fedora-specific setup script
chmod +x app/setup.sh
./app/setup.sh
```

The setup script will:

- Install required system dependencies (PostgreSQL dev libraries, Python dev tools)
- Create a Python virtual environment
- Install all Python dependencies
- Set up environment variables

If the script fails, manually install dependencies:

```bash
sudo dnf install postgresql-devel python3-devel gcc python3-pip python3-virtualenv red
```

Step 2: Google Cloud Services Setup (15 minutes)

2.1 Create Google Cloud Project

1. Go to Google Cloud Console
2. Click **"New Project"** → Name it "fionntan-podcast" → **"Create"**
3. Select your new project from the dropdown

2.2 Enable Required APIs

Go to APIs & Services > Library and enable:

- **Cloud Text-to-Speech API**
- **Cloud Storage API**
- **Generative Language API** (for Gemini)

2.3 Create Service Account

1. Go to IAM & Admin > Service Accounts
2. Click **"Create Service Account"**
3. Name: `fionntan-generator`
4. Grant these roles:
   - **Cloud Text-to-Speech Admin**
   - **Storage Admin**
   - **AI Platform Developer**
5. Click **"Create and Continue"** → **"Done"**

2.4 Download Credentials

1. Click on your service account email
2. Go to **"Keys"** tab → **"Add Key"** → **"Create new key"**
3. Select **JSON** → **"Create"**
4. Save the downloaded file as `credentials/service-account.json` in your project

2.5 Create Storage Bucket

1. Go to Cloud Storage
2. Click **"Create Bucket"**
3. Name: `your-name-fionntan-storage` (must be globally unique)
4. Choose your nearest region
5. **Access control**: Uniform
6. Click **"Create"**

2.6 Get Gemini API Key

1. Go to Google AI Studio
2. Click **"Create API Key"**
3. Copy the generated key

Step 3: Environment Configuration (5 minutes)

Create .env File

In your project root, create `.env`:

```env
# Google Cloud Services
GOOGLE_APPLICATION_CREDENTIALS=./credentials/service-account.json
GCP_PROJECT_ID=your-project-id
GCS_BUCKET_NAME=your-bucket-name
GEMINI_API_KEY=your-gemini-api-key

# Celery/Redis
CELERY_BROKER_URL=redis://localhost:6379/0
CELERY_RESULT_BACKEND=redis://localhost:6379/0

# Flask
SECRET_KEY=your-secure-random-key-here
FLASK_ENV=development
DATABASE_URL=sqlite:///app.db
```

Replace these values:

- `your-project-id`: From Google Cloud Console top bar
- `your-bucket-name`: The bucket you created
- `your-gemini-api-key`: From Google AI Studio
- `your-secure-random-key-here`: Any long random string

Verify Credentials Setup

```bash
# Create credentials directory
mkdir -p credentials

# Move your downloaded JSON file to the right location
mv ~/Downloads/your-project-***.json ./credentials/service-account.json

# Verify file exists
ls -la ./credentials/service-account.json
```

Step 4: Test Services (10 minutes)

Activate Environment and Test

```bash
# Activate the virtual environment created by setup.sh
source venv/bin/activate

# Test all services
python test_all_services.py
```

Expected output:

```
🧪 TEXT-TO-SPEECH SERVICE
✅ TTS Client initialization
✅ Audio generation (XXXX bytes generated)

🧪 CLOUD STORAGE SERVICE
✅ Storage client initialization
✅ Bucket access
✅ File upload
✅ File download
✅ File cleanup

🧪 GEMINI API SERVICE
✅ Gemini client initialization
✅ Script generation

🧪 END-TO-END WORKFLOW TEST
✅ Paper retrieval
✅ Script generation
✅ Audio generation
✅ Audio storage
✅ Workflow cleanup

🎉 END-TO-END WORKFLOW SUCCESSFUL!
```

If any service fails, check:

- Credentials file path and permissions
- API keys in .env file
- Bucket name and permissions
- Internet connectivity

Step 5: Start the Application (5 minutes)

Terminal Setup

You'll need 3 terminals running simultaneously:

Terminal 1 - Redis:

```bash
redis-server
```

Terminal 2 - Celery Worker:

```bash
cd fionntan
source venv/bin/activate
export $(cat .env | xargs)
python celery_worker.py
```

Terminal 3 - Flask API:

```bash
cd fionntan
source venv/bin/activate
python main.py
```

Verify Everything is Running

- Redis should show: `"Ready to accept connections"`
- Celery should show: `"celery@hostname ready"`
- Flask should show: `"Running on http://127.0.0.1:5000"`

Step 6: Disable Authentication (For Testing)

Edit `app/api/podcasts.py`:

python

```python
# Find these functions and comment out @jwt_required() decorators:

# @jwt_required()  # <-- Comment this line out
def create_podcast():
    """Create a new podcast generation task."""
    try:
        # user_id = get_jwt_identity()  # <-- Comment this out
        user_id = 1  # <-- Add this line
        # ... rest of function stays the same

# @jwt_required()  # <-- Comment this line out
def get_podcast(podcast_id):
    """Get podcast details."""
    try:
        # user_id = get_jwt_identity()  # <-- Comment this out
        user_id = 1  # <-- Add this line
        # ... rest of function stays the same

# @jwt_required()  # <-- Comment this line out
def get_podcast_audio(podcast_id):
    """Stream or download podcast audio."""
    try:
        # user_id = get_jwt_identity()  # <-- Comment this out
        user_id = 1  # <-- Add this line
        # ... rest of function stays the same
```

Restart Flask (Terminal 3):

bash

```bash
# Stop with Ctrl+C, then restart:
python main.py
```

Step 7: Generate Physics Podcast Test! (2 minutes)

Create the Fractional Quantum Hall Effect Podcast

```bash
curl -X POST http://localhost:5000/api/v1/podcasts \
  -H "Content-Type: application/json" \
  -d '{
    "title": "Breakthrough in Fractional Quantum Hall Physics",
    "technical_level": "advanced",
    "target_length": 12,
    "use_preferences": false,
    "paper_ids": ["2308.02657", "2309.17436"]
  }'
```

Expected response:

```json
{
  "podcast_id": 1,
  "task_id": "abc-123-def",
  "status": "queued",
  "created_at": "2025-06-04T..."
}
```

Monitor Generation Progress

Watch Terminal 2 (Celery) for progress:

```
[INFO] Task received
[INFO] Fetching paper: 2308.02657
[INFO] Successfully fetched: Observation of Fractionally Quantized...
[INFO] Fetching paper: 2309.17436
[INFO] Successfully fetched: Fractional Quantum Anomalous Hall Effect...
[INFO] Successfully generated podcast script
[INFO] Successfully synthesized speech (4MB audio)
[INFO] Successfully uploaded audio to storage
```

Download and Listen

```bash
# Check podcast status (use the podcast_id from your response)
curl http://localhost:5000/api/v1/podcasts/1

# Download the generated podcast
curl http://localhost:5000/api/v1/podcasts/1/audio > quantum_hall_podcast.mp3

# Check file size (should be ~4MB for 12 minutes)
ls -lh quantum_hall_podcast.mp3

# Listen to your physics podcast!
mpv quantum_hall_podcast.mp3
# or
vlc quantum_hall_podcast.mp3
```

🎉 Success Criteria

You should now have:

- ✅ A 10-12 minute AI-generated podcast
- ✅ Two AI hosts discussing breakthrough quantum physics papers
- ✅ High-quality TTS narration
- ✅ Clear discussion of fractional quantum Hall effects
- ✅ Professional audio quality

Troubleshooting

If ArXiv papers fail to fetch:

```bash
# Test individual paper access:
python -c "
from app import create_app
from app.services.arxiv_service import ArxivService
app = create_app('development')
with app.app_context():
    service = ArxivService()
    paper = service.get_paper_by_id('2308.02657')
    print('✅ Paper found:' if paper else '❌ Paper failed:', paper['title'][:50] if pa
"
```

If Celery worker fails:

```bash
# Check Redis is running:
redis-cli ping  # Should return "PONG"

# Restart Celery with debug info:
celery -A app.celery worker --loglevel=debug
```

If Google Cloud APIs fail:

```bash
# Test credentials:
python -c "
from google.cloud import storage
client = storage.Client()
print('✅ Google Cloud authenticated successfully')
"
```

If audio generation fails:

- Check your Google Cloud billing is enabled
- Verify TTS API quotas aren't exceeded
- Ensure bucket permissions are correct

Next Steps

Once this works, you can:

- **Test with other topics**: Change the paper IDs to any ArXiv papers
- **Adjust technical level**: Use "beginner", "intermediate", or "advanced"
- **Create longer podcasts**: Increase target_length up to 60 minutes
- **Enable authentication**: Set up Google OAuth for multi-user access
- **Deploy to production**: Use the included Docker setup

Support

If you encounter issues:

1. Check that all 3 terminals are running without errors
2. Verify your .env file has all required values
3. Test each service individually using the test script
4. Check Google Cloud Console for API quota limits

Your AI podcast generator is now fully operational! 🎙️🎉