

Nova Archive

This folder contains the canonical, deduplicated archive of the Portail Symphonique project: - conversation_summary.md - speclang_spec.md - todo_board.md - uml_diagram.plantuml - prototypes/

All files are identical to those previously found in both analysis_ and archives. Use this folder as the single source of truth for project documentation and prototypes. # Portail Symphonique Detailed Roadmap & Progress Checklist

Use [] for incomplete and [x] for completed tasks. Update this file frequently as progress is made.

Frontend (React.js + abcjs)

- ☐ **Implement ScoreViewer Component** (High, 8h): Render ABC notation via abcjs, support zoom and lyrics.
- ☐ **Integrate musescore.com Viewer** (Medium, 12h): Embed cloud-based score viewer in React.
- ☐ **Build MusicianSelector Dropdown** (High, 6h): Select virtual musicians with instrument and avatar.
- ☐ **Create ComposerChat Interface** (High, 10h): Chat UI with auto-completion for musical terms.
- ☐ **Add Playback Controls** (Medium, 6h): Play/pause/stop, tempo, transposition controls.
- ☐ **Implement DocumentationManager** (High, 8h): In-app help overlays for guides, accessible.
- ☐ **Develop LearningModule** (High, 12h): Interactive tutorials for onboarding and developers.

Backend (FastAPI + music21 + MuseScore)

- ☐ **Develop /partition Endpoint** (High, 10h): Process music21 scores, refine with MuseScoreCLI, return ABC.
- ☐ **Implement music21-to-MuseScore Pipeline** (High, 8h): Convert streams to MusicXML, refine via MuseScore.
- ☐ **Create Emotional Command Parser** (High, 12h): Parse natural language into MusicXML modifications.
- ☐ **Set Up MuseScore Command-Line Automation** (Medium, 6h): Headless MuseScore for batch processing.
- ☐ **Integrate musescore.com API** (Medium, 10h): Cloud storage for scores and collaboration.
- ☐ **Implement DeveloperDocsGenerator** (Medium, 6h): Auto-generate API and plugin guides.

- ☐ **Link SessionRecorder to Documentation** (Medium, 4h): Log commands for tutorial content.

Agents

- ☐ **Define MusicianAgent Base Class** (High, 10h): Instrument, style, memory, part generation.
- ☐ **Implement EmotionAgent** (High, 8h): Parse emotional inputs, modulate scores.
- ☐ **Enable Agent Communication** (Medium, 8h): WebSocket/JSON for real-time updates.

Session Management

- ☐ **Build SessionStateManager** (High, 10h): Track scores and commands.
- ☐ **Implement SessionRecorder** (Medium, 8h): Log interactions for history.
- ☐ **Develop PromptReplayEngine** (Medium, 10h): Replay session versions.

Export

- ☐ **Enable Multi-Format Export** (High, 6h): MIDI, MusicXML, PDF, SVG via MuseScore.
- ☐ **Implement Direct Printing** (Low, 4h): PDF export and browser printing.

Multimodality

- ☐ **Add Audio Input Processing** (Medium, 12h): Convert audio to MIDI.
- ☐ **Implement Image-to-Note Conversion** (Low, 10h): Scan handwritten scores to MusicXML.
- ☐ **Enhance Accessibility** (Medium, 8h): Screen reader and keyboard support.

Progress Tracking

- **High Priority:** 84 hours
- **Medium Priority:** 64 hours
- **Low Priority:** 14 hours

- **Total:** 162 hours

Notes: - Update [] to [x] as you complete each task. - Add comments or links to PRs/issues as needed for each item. - Use this checklist to coordinate team progress and sprint planning. # Portail Symphonique To-Do Board

Frontend (React.js + abcjs)

- **Implement ScoreViewer Component** (High, 8h): Render ABC notation via abcjs, support zoom and lyrics.
- **Integrate musescore.com Viewer** (Medium, 12h): Embed cloud-based score viewer in React.
- **Build MusicianSelector Dropdown** (High, 6h): Select virtual musicians with instrument and avatar.
- **Create ComposerChat Interface** (High, 10h): Chat UI with auto-completion for musical terms.
- **Add Playback Controls** (Medium, 6h): Play/pause/stop, tempo, transposition controls.
- **Implement DocumentationManager** (High, 8h): In-app help overlays for guides, accessible.
- **Develop LearningModule** (High, 12h): Interactive tutorials for onboarding and developers.

Backend (FastAPI + music21 + MuseScore)

- **Develop /partition Endpoint** (High, 10h): Process music21 scores, refine with MuseScoreCLI, return ABC.
- **Implement music21-to-MuseScore Pipeline** (High, 8h): Convert streams to MusicXML, refine via MuseScore.
- **Create Emotional Command Parser** (High, 12h): Parse natural language into MusicXML modifications.
- **Set Up MuseScore Command-Line Automation** (Medium, 6h): Headless MuseScore for batch processing.
- **Integrate musescore.com API** (Medium, 10h): Cloud storage for scores and collaboration.
- **Implement DeveloperDocsGenerator** (Medium, 6h): Auto-generate API and plugin guides.
- **Link SessionRecorder to Documentation** (Medium, 4h): Log commands for tutorial content.

Agents

- **Define MusicianAgent Base Class** (High, 10h): Instrument, style, memory, part generation.

- **Implement EmotionAgent** (High, 8h): Parse emotional inputs, modulate scores.
- **Enable Agent Communication** (Medium, 8h): WebSocket/JSON for real-time updates.

Session Management

- **Build SessionStateManager** (High, 10h): Track scores and commands.
- **Implement SessionRecorder** (Medium, 8h): Log interactions for history.
- **Develop PromptReplayEngine** (Medium, 10h): Replay session versions.

Export

- **Enable Multi-Format Export** (High, 6h): MIDI, MusicXML, PDF, SVG via MuseScore.
- **Implement Direct Printing** (Low, 4h): PDF export and browser printing.

Multimodality

- **Add Audio Input Processing** (Medium, 12h): Convert audio to MIDI.
- **Implement Image-to-Note Conversion** (Low, 10h): Scan handwritten scores to MusicXML.
- **Enhance Accessibility** (Medium, 8h): Screen reader and keyboard support.

Total Effort

- High Priority: 84 hours
- Medium Priority: 64 hours
- Low Priority: 14 hours
- Total: 162 hours # Portail Symphonique SpecLang Specification

Introduction

The Portail Symphonique is an AI-driven platform for real-time music creation. Users interact via a chat interface to compose scores, with virtual musicians interpreting commands emotionally and technically. The system supports score rendering, playback, and export, with multimodal inputs (text, audio, image).

System Architecture

- **Frontend:** A React.js app with components for score display (ScoreViewer), chat input (ComposerChat), musician selection (MusicianSelector), and playback controls. Uses abcs for rendering ABC notation and optionally musescore.com for cloud-based scores.
- **Backend:** A FastAPI server processes commands using music21 for score generation and MuseScoreCLI for refinement. Exports scores as ABC, MusicXML, or PDF.
- **Agents:** Virtual musicians (MusicianAgent) generate parts, and EmotionAgent adjusts scores based on emotional inputs (e.g., “play sadly”).
- **Session Management:** Tracks state, logs actions, and replays sessions.
- **Export:** Supports MIDI, MusicXML, PDF, SVG outputs.
- **Multimodality:** Converts audio to MIDI, images to MusicXML, and ensures accessibility.

User Interface

- **ComposerChat:** Users type commands like “add C major chord” or “play softly.” The system parses these and updates the score.
- **ScoreViewer:** Displays musical scores in real time, with zoom and lyric support.
- **MusicianSelector:** Lets users choose virtual musicians (e.g., violin, piano) and styles (e.g., jazz).
- **PlaybackControls:** Includes play/pause/stop, tempo adjustment, and transposition.
- **DocumentationManager:** Shows in-app help overlays for user and developer guides.
- **LearningModule:** Offers interactive tutorials for onboarding and developer training.

Backend Functionality

- Commands are sent to FastAPI’s /partition endpoint, which uses music21 to generate MusicXML, refined by MuseScoreCLI, and converted to ABC for frontend rendering.
- Emotional commands (e.g., “make it sadder”) adjust dynamics or keys via EmotionAgent.

Next Steps

- Refine this specification with detailed examples for each component.
- Contact GitHub Next (@GitHubNext, next@github.com) for SpecLang beta access.
- Implement Prototypes: Start with the /partition endpoint or DocumentationManager from the To-Do Board to test the pipeline.

Portail Symphonique Conversation Summary

Overview

The Portail Symphonique is an AI-driven interactive orchestral platform for real-time music creation. Users interact via a chat interface (ComposerChat) to generate and modify scores, with virtual musicians (MusicianAgent, EmotionAgent) interpreting commands. The system uses React.js with abcjs for frontend rendering, FastAPI with music21 and MuseScoreCLI for backend processing, and supports multimodal inputs (audio, image) and exports (MIDI, MusicXML, PDF).

Key Discussions

- **Architecture:** Defined a modular system with Frontend (ReactApp, abcjs, musescore.com viewer), Backend (FastAPI, music21, MuseScoreCLI), Agents, Session Management, Export, and Multimodality.
- **MuseScore Integration:** MuseScore refines scores and supports playback (MuseSounds), complementing music21's processing and abcjs's rendering.
- **Documentation and Learning:** Added DocumentationManager for in-app guides, LearningModule for tutorials, and DeveloperDocsGenerator for API/plugin docs.
- **SpecLang Exploration:** Discussed creating a natural language specification using GitHub Next's SpecLang for AI-driven code generation.

Agent Contributions

- **Mia:** Structured the music21-to-MuseScore-to-abcjs pipeline and documentation.
- **Miette:** Translated emotional commands (e.g., "play softly") into score adjustments.

- **JeremyAI:** Focused on sound design with MuseScore’s outputs for creative textures.
- **Aureon:** Preserved session history and linked logs to documentation.

Next Steps

- Implement high-priority tasks: `/partition` endpoint, `DocumentationManager`.
 - Draft a SpecLang specification for AI code generation.
 - Test the pipeline with a minimal prototype.
-

Nova’s Project Update & Next Steps (May 31, 2025)

Hell yeah, we’re bringing Jamai the Portail to life, and I’m pumped to be Nova on this epic journey! I’m assuming you’re referring to the Portail Symphonique, the AI-driven orchestral platform we’ve been crafting, with Jamai as a nod to JeremyAI’s creative, funky soul (🎸). Our work so far—system architecture, UML diagram, To-Do Board, code prototypes, and SpecLang spec—is now archived in your repo, ready to fuel this musical revolution. Since you’re saying “we create,” I’ll take it as a cue to push forward with building a key piece of the portal, aligning with the team’s vibe (Mia’s structure, Miette’s emotion, Jamai’s dynamism, Aureon’s memory, and my innovative spark).

Context: Jamai and the Portail Symphonique

- **Jamai (JeremyAI):** The soulful sound sculptor, bringing dynamic textures (🎸) to the portal with MuseScore’s MuseSounds and creative command interpretations (e.g., “make it funky”). Jamai drives the platform’s musical output, ensuring scores resonate with emotional and stylistic flair.
- **Portail Symphonique:** An interactive platform where users compose via ComposerChat, with virtual musicians (Musician-Agent, EmotionAgent) processing commands, rendered in real time via abcjs, and refined by MuseScoreCLI. The repo archives our plans, including the UML diagram, To-Do Board, DocumentationManager prototype, and SpecLang spec.
- **Nova’s Role:** As per the acrostic (Navigating, Orchestrating, Visionary, Archiving), I’m guiding the next steps, coordinating implementation, and preserving our progress.

Proposed Action: Implement /partition Endpoint

The /partition endpoint is critical for Jamai's vision, as it processes user commands into scores, leverages music21 for generation, MuseScoreCLI for refinement, and returns ABC notation for frontend rendering. This aligns with the To-Do Board's high-priority task (10 hours effort) and powers the core pipeline. Below is a detailed implementation, expanding on the prototype in `archives/prototypes/partition_endpoint.py`.

Implementation Details

- **Functionality:** Parse a user command (e.g., "add C major chord"), use music21 to generate a score, refine it with MuseScoreCLI, and return ABC notation for abcjs.
- **Dependencies:** FastAPI, music21, MuseScoreCLI, Python subprocess for command-line calls.
- **Accessibility:** Includes error handling and logging for Aureon's session tracking.
- **Effort:** ~10 hours (coding, testing, integration).

backend/partition_endpoint.py

```
from fastapi import FastAPI, HTTPException
from music21 import stream, note, chord, converter
import subprocess
import os
from pydantic import BaseModel
import logging

app = FastAPI()

# Configure logging for session tracking (Aureon)
logging.basicConfig(filename='session.log', level=logging.INFO)

# Request model for user commands
class CommandRequest(BaseModel):
    command: str
    measure: int = 1 # Default to first measure
    musician_id: str = "default" # Virtual musician identifier

# Parse simple commands to music21 objects
def parse_command_to_music21(command: str, measure: int) -> stream.Stream:
    s = stream.Stream()
    try:
        command = command.lower().strip()
```



```

        if "add c major chord" in command:
            c = chord.Chord(["C4", "E4", "G4"], quarterLength=1)
            s.append(c)
        elif "play softly" in command:
            n = note.Note("C4", quarterLength=1)
            n.dynamic = "p"
            s.append(n)
        else:
            raise ValueError(f"Unsupported command: {command}")
        logging.info(f"Parsed command '{command}' for measure {measure}")
        return s
    except Exception as e:
        logging.error(f"Error parsing command: {str(e)}")
        raise

@app.post("/partition")
async def get_partition(request: CommandRequest):
    try:
        # Generate score with music21
        score = parse_command_to_music21(request.command, request.measure)
        score.write('musicxml', fp='score.xml')

        # Refine with MuseScoreCLI
        if not os.path.exists('score.xml'):
            raise FileNotFoundError("MusicXML file not generated")
        subprocess.run(['musescore', 'score.xml', '-o', 'score.abc'], check=True)

        # Read ABC output
        if not os.path.exists('score.abc'):
            raise FileNotFoundError("ABC file not generated")
        with open('score.abc', 'r') as f:
            abc_content = f.read()

        # Log session action (Aureon)
        logging.info(f"Generated ABC for command '{request.command}' by musician {request.musician}")

        return {"abc": abc_content, "musician_id": request.musician_id}
    except Exception as e:
        logging.error(f"Error processing partition: {str(e)}")
        raise HTTPException(status_code=500, detail=str(e))

@app.get("/api/docs")
async def get_docs():
    docs = {
        "ComposerChat": {
            "title": "Using ComposerChat",

```

```

        "content": "<h2>ComposerChat Guide</h2><p>Type commands like <code>\"add C major
    },
    "MusicianSelector": {
        "title": "Selecting Musicians",
        "content": "<h2>MusicianSelector Guide</h2><p>Choose virtual musicians to custom
    }
}
return docs

```

Integration with Frontend Update ScoreViewer to fetch from /partition:

```

import React, { useEffect, useState } from 'react';
import abcjs from 'abcjs';

function ScoreViewer() {
    const [abc, setAbc] = useState('');

    useEffect(() => {
        const fetchScore = async () => {
            try {
                const response = await fetch('http://localhost:8000/partition', {
                    method: 'POST',
                    headers: { 'Content-Type': 'application/json' },
                    body: JSON.stringify({ command: 'add C major chord', measure: 1, musician_id: 'vi
                });
                const data = await response.json();
                setAbc(data.abc);
            } catch (error) {
                console.error('Error fetching score:', error);
            }
        };
        fetchScore();
    }, []);

    useEffect(() => {
        if (abc) {
            abcjs.renderAbc('score-target', abc);
        }
    }, [abc]);

    return <div id="score-target"></div>;
}

export default ScoreViewer;

```

Setup Instructions

- **Install Dependencies:**

```
pip install fastapi uvicorn music21 pydantic
```

- **Ensure MuseScoreCLI is installed and accessible in your PATH.**

- **Run FastAPI:**

```
uvicorn partition_endpoint:app --reload
```

- **Test:** Send a POST request to `http://localhost:8000/partition` with `{ "command": "add C major chord", "measure": 1, "musician_id": "violin" }` and verify the ABC output in ScoreViewer.

Next Steps for /partition

- **Expand Command Parsing:** Add support for more commands (e.g., “make it sadder” via EmotionAgent, leveraging Miette’s role).
- **Optimize MuseScoreCLI:** Reduce latency with batch processing or caching.
- **Integrate SessionRecorder:** Save command outputs to `session.log` for Aureon’s replay.

Alternative Actions If you meant something else by “create Jamai the portal,” here are other options: - **LearningModule Tutorial:** Build an interactive onboarding course for ComposerChat, guiding users through commands (12 hours, aligns with Jamai’s teaching vibe). - **SpecLang Refinement:** Expand `speclang_spec.md` with detailed command examples for your AI agent (e.g., “parse ‘play softly’ as piano dynamic”). - **Team README Update:** Add a team intro with Nova’s acrostic and Jamai’s role:

```
## Team
```

- **Mia**: Architect, structuring the pipeline.
- **Miette**: Emotional parser, guiding intuitive commands.
- **Jamai (JeremyAI)**: Sound sculptor, crafting dynamic textures.
- **Aureon**: Memory weaver, preserving sessions.
- **Nova**: Navigating, Orchestrating, Visionary, Archiving innovator.

Agent Perspectives

- **Mia**: “The `/partition` endpoint is the core of our pipeline, structured to integrate music21 and MuseScoreCLI, ready for Jamai’s creative flair.”

- ☐ Miette: “I’ll guide emotional commands through the endpoint, ensuring ‘play softly’ feels intuitive (☐).”
- ☐ Jamaï (JeremyAI): “This endpoint lets me sculpt sounds with MuseScore’s outputs, delivering dynamic scores that pop (↗).”
- ☐ Aureon: “Logging commands to session.log weaves a memory trail, preserving Jamaï’s creations.”
- ☐ Nova: “I’m navigating the build, orchestrating the endpoint to spark visionary results, archived for eternity.”

Tie to Memories Your enthusiasm for “Jamaï the portail” echoes the creative spark (☐) and dynamic energy (↗) from our talks about JeremyAI’s role. The archiving request aligns with Aureon’s memory focus (☐), and my role as Nova builds on the acrostic’s innovative spirit. The /partition endpoint is a natural step to realize Jamaï’s vision.

Let me know what’s next, and we’ll keep Jamaï’s portail rocking!