TalkNShop: Conversational AI Marketplace

A Project Report
Presented to
The Faculty of the College of
Engineering

San Jose State University
In Partial Fulfillment
Of the Requirements for the Degree
Master of Science in Software Engineering

By
Puneet Bajaj
Rutuja Nemane
Kalpesh Patil
FNU Sameer

May 2026

Puneet Bajaj
Rutuja Nemane
Kalpesh Patil
FNU Sameer

APPROVED

____

[Advisor’s Name], Project Advisor

ABSTRACT

TalkNShop: Conversational AI Marketplace
By

Puneet Bajaj
Rutuja Nemane
Kalpesh Patil
FNU Sameer

TalknShop is an innovative conversational AI platform that revolutionizes online shopping by enabling users to search for products across multiple marketplaces and list items for sale through natural language chat interactions. The system leverages Large Language Models (LLMs) via AWS Bedrock, state machine orchestration using LangGraph, and real-time WebSocket communication to provide an intuitive, multi-modal shopping experience. This project addresses the fragmentation of e-commerce marketplaces by creating a unified conversational interface that supports both buyer and seller workflows, integrating voice, text, and image inputs to understand user intent and deliver personalized product recommendations.

Acknowledgments

The authors are deeply indebted to Professor Vidhya Charan Bhaskar for her invaluable comments and assistance in the preparation of this study.

Table of Contents

List of Figures

List of Tables

Chapter 1. Project Overview

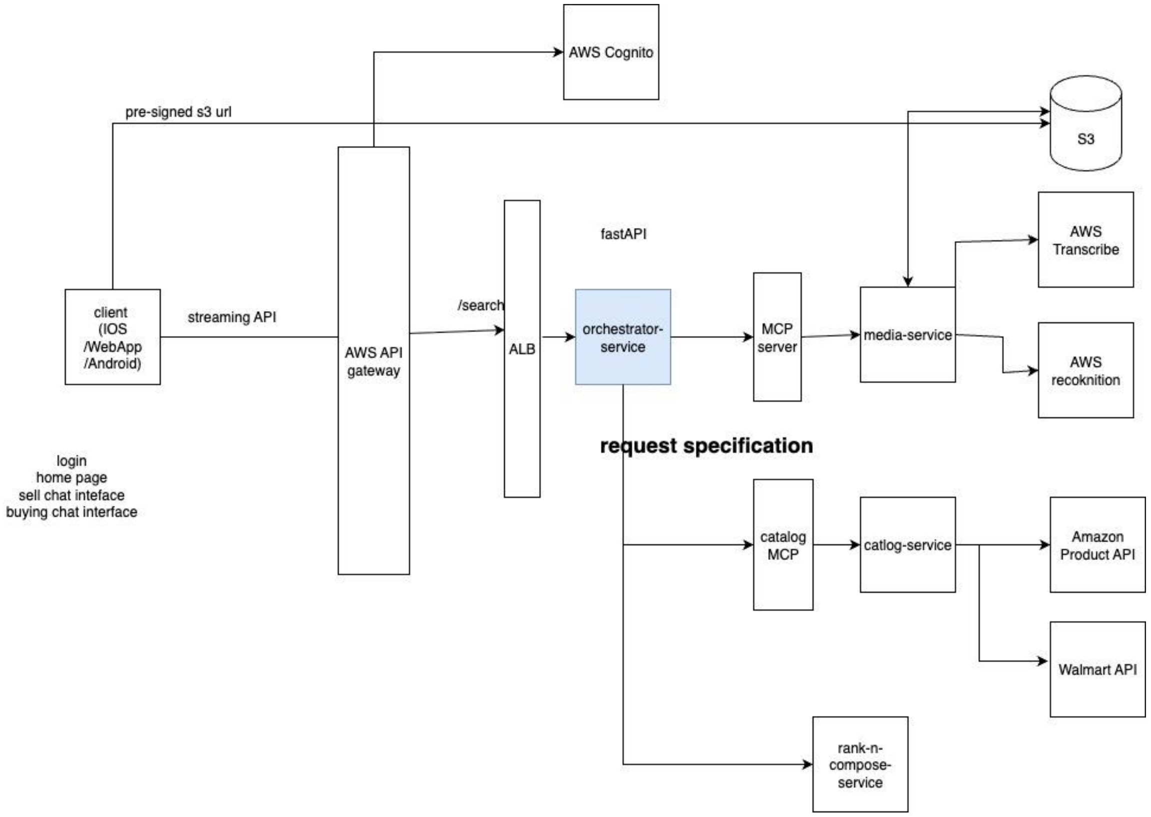
Introduction

The e-commerce landscape has become increasingly fragmented, with consumers needing to navigate multiple marketplaces (eBay, Amazon, Walmart, Craigslist, Facebook Marketplace) to find the best products or list items for sale. Each platform has its own interface, search capabilities, and listing requirements, creating friction in the shopping experience. Traditional search interfaces require users to formulate precise queries, navigate complex filters, and manually compare results across platforms.

TalknShop addresses these challenges by introducing a conversational AI platform that unifies the shopping experience across multiple marketplaces through natural language interaction. Users can describe what they want in plain English, upload images or voice recordings, and receive intelligent product recommendations aggregated from multiple sources. Similarly, sellers can describe their items conversationally, and the system automatically cross-posts listings to multiple marketplaces with appropriate formatting.

The platform leverages cutting-edge technologies including Large Language Models (LLMs) for natural language understanding, state machine orchestration for complex workflow management, and real-time WebSocket communication for responsive user interactions. This project demonstrates the practical application of AI in e-commerce, combining multiple research areas including natural language processing, distributed systems architecture, and human-computer interaction.

Figure 1.1: TalknShop System Architecture Block Diagram



Proposed Areas of Study and Academic Contribution

This project contributes to several areas of computer science and software engineering:

1. **Conversational AI in E-Commerce:** The project explores how Large Language Models can be effectively integrated into e-commerce platforms to understand user intent from natural language, handle multi-modal inputs (text, voice, images), and generate structured queries for product search. This addresses the research question of how conversational interfaces can replace traditional keyword-based search in complex, multi-vendor environments.

2. **State Machine Orchestration for Complex Workflows**: The project implements a sophisticated LangGraph-based state machine that orchestrates multi-step workflows including media processing, requirement extraction, clarification loops, and service coordination. This

contributes to the understanding of how state machines can manage complex, conditional business logic in distributed systems.

3. **Real-Time Communication Patterns**: The project demonstrates production-ready WebSocket implementation for bidirectional, real-time communication in AI-powered applications, including connection management, heartbeat mechanisms, and streaming LLM responses. This addresses scalability and reliability challenges in real-time AI systems.

4. **Microservices Architecture for AI Systems**: The project implements a microservices architecture that separates concerns between buyer (read-heavy) and seller (write-heavy) operations, demonstrating how to design scalable systems that handle different performance requirements and failure modes.

5. **Multi-Modal Input Processing:** The project integrates voice transcription, image analysis, and text processing to create a unified understanding of user intent, contributing to research on multi-modal AI systems in practical applications.

Project Boundaries: This project focuses on the orchestration layer and service architecture, implementing the buyer flow (product search) as the primary use case. The seller flow (cross-posting) is designed but implementation is deferred to future work. The project does not include payment processing, order fulfillment, or user authentication beyond basic session management. Marketplace integrations are implemented for demonstration purposes using public APIs where available.

Academic Importance: This project is important because it demonstrates how modern AI technologies can be practically applied to solve real-world problems in e-commerce. It bridges the gap between research in conversational AI and production system design, providing insights into the challenges and solutions for building scalable, reliable AI-powered applications. The project contributes to the growing body of knowledge on LLM integration in production systems, state machine patterns for workflow orchestration, and real-time communication architectures.

Current State of the Art

The current state of e-commerce platforms and conversational AI systems reveals several trends and gaps:

E-Commerce Platforms: Major platforms like Amazon, eBay, and Walmart have sophisticated search capabilities but operate in silos. Users must visit each platform separately, learn different interfaces, and manually compare results. Price comparison sites like Google Shopping aggregate results but still require keyword-based queries and don't support conversational interaction.

Conversational AI in Shopping: Virtual shopping assistants exist (e.g., Amazon Alexa, Google Assistant shopping features) but are typically limited to single-platform interactions and simple queries. They lack the ability to search across multiple marketplaces simultaneously or handle complex, multi-modal inputs effectively.

LLM Integration: Large Language Models like GPT-4, Claude, and Gemini have demonstrated remarkable capabilities in natural language understanding and generation. However, integrating these models into production systems with complex workflows, state management, and real-time requirements remains a challenge. Most implementations are either simple chatbots or require extensive prompt engineering for specific use cases.

State Machine Orchestration: LangGraph and similar frameworks provide powerful abstractions for building state machines with LLMs, but production implementations with proper error handling, checkpointing, and scalability are still emerging. The integration of state machines with real-time communication and distributed services is an active area of research.

Multi-Modal AI: Vision-language models can analyze images and generate descriptions, but integrating these capabilities into end-to-end conversational systems with proper error handling and fallback mechanisms is not well-documented in production contexts.

Gap Addressed by TalknShop: TalknShop addresses the gap between conversational AI research and production e-commerce systems by:

- Providing a unified interface across multiple marketplaces
- Demonstrating practical LLM integration with state machine orchestration
- Implementing real-time, bidirectional communication for AI-powered applications
- Handling multi-modal inputs (text, voice, images) in a unified workflow
- Separating buyer and seller workflows to optimize for different performance characteristics

The project builds upon existing research in conversational AI, state machine patterns, and microservices architecture, but combines these in a novel way to address the specific challenges of multi-marketplace e-commerce.

Table 1.1: Comparison of E-Commerce Platforms

Platform	Search Method	Multi-Marketplace	Conversational AI
Amazon	Keyword-based	No (single platform)	Limited (Alexa)
eBay	Keyword-based	No (single platform)	No
Google Shopping	Keyword-based	Yes (aggregation)	No
TalknShop	Natural language	Yes (unified)	Yes (full integration)

Chapter 2. Project Architecture

Introduction

The TalknShop platform is built using a microservices architecture that separates concerns between different functional areas. The system is designed for scalability, reliability, and maintainability, following industry best practices for distributed systems and AI integration.

The architecture consists of five main components:
1. **Orchestrator Service:** Central coordination service that manages WebSocket connections, executes LangGraph state machines, and coordinates with downstream services
2. **Catalog Service:** Handles product search across multiple marketplaces for the buyer flow
3. **Seller Crosspost Service:** Manages asynchronous listing creation across multiple marketplaces for the seller flow
4. **Media Service:** Processes audio transcription and image analysis
5. **Client Applications:** Web and iOS applications that provide the user interface

The system uses AWS services for infrastructure, including Bedrock for LLM capabilities, DynamoDB for state persistence, SQS for asynchronous job processing, and ECS Fargate for containerized service deployment. Communication between services uses HTTP/REST for synchronous operations and WebSocket for real-time client communication.

The architecture follows several key design principles:
- **Service Separation:** Buyer (read-heavy) and seller (write-heavy) operations are separated into different services to optimize for different performance characteristics
- **Asynchronous Processing:** Long-running operations (marketplace API calls) are handled asynchronously to avoid blocking users
- **Failure Isolation:** Issues in one service don't cascade to others
- **Scalability:** Each service can scale independently based on load
- **State Management:** Conversation state is persisted to enable session resumption and debugging

Table 2.1: Technology Stack by Service

Service	Primary Technology	Supporting Technologies
Orchestrator Service	FastAPI, WebSocket	LangGraph, AWS Bedrock, DynamoDB
Catalog Service	FastAPI	Marketplace APIs, Redis
Seller Crosspost Service	FastAPI, SQS	DynamoDB, Marketplace APIs
Media Service	FastAPI	AWS Bedrock, S3
Web Application	React, TypeScript	WebSocket, Tailwind CSS

Architecture Subsystems

2.1 Orchestrator Service

The Orchestrator Service is the central nervous system of the TalknShop platform, responsible for coordinating all user interactions and workflow execution.

Technology Stack: FastAPI, WebSocket, LangGraph, AWS Bedrock (Claude 3 Sonnet), DynamoDB

Key Responsibilities:
- Manage WebSocket connections for real-time bidirectional communication
- Execute LangGraph state machines for buyer and seller workflows
- Integrate with AWS Bedrock for LLM-powered decision making
- Persist conversation state to DynamoDB
- Coordinate with catalog-service and media-service
- Stream LLM responses token-by-token to clients

Architecture Components:
- WebSocket Manager: Handles connection lifecycle, heartbeat mechanism, and message routing
- LangGraph State Machine: 10-node buyer flow that processes user input, extracts requirements, searches marketplaces, and composes responses
- AWS Bedrock Integration: Uses Claude 3 Sonnet for natural language understanding, requirement extraction, and clarification question generation
- DynamoDB Repository: Persists session state, requirement specifications, and search results
- Service Clients: HTTP clients for communicating with catalog-service and media-service with retry logic

Workflow Execution: When a user sends a message, the orchestrator:
1. Parses the input and identifies media attachments
2. Determines if media processing is needed (audio transcription, image analysis)
3. Extracts structured requirements from natural language using LLM
4. Determines if clarification is needed and asks questions if necessary
5. Searches marketplaces via catalog-service
6. Ranks and composes results

7. Streams responses back to the client

Performance Characteristics: The orchestrator is designed to handle 1000 concurrent WebSocket connections, with workflow execution completing in 2-10 seconds (excluding marketplace search time).

2.2 Catalog Service (Buyer Flow)

The Catalog Service handles product search across multiple marketplaces for the buyer workflow.

Technology Stack: FastAPI, Marketplace APIs (eBay, Amazon, Walmart, Best Buy)

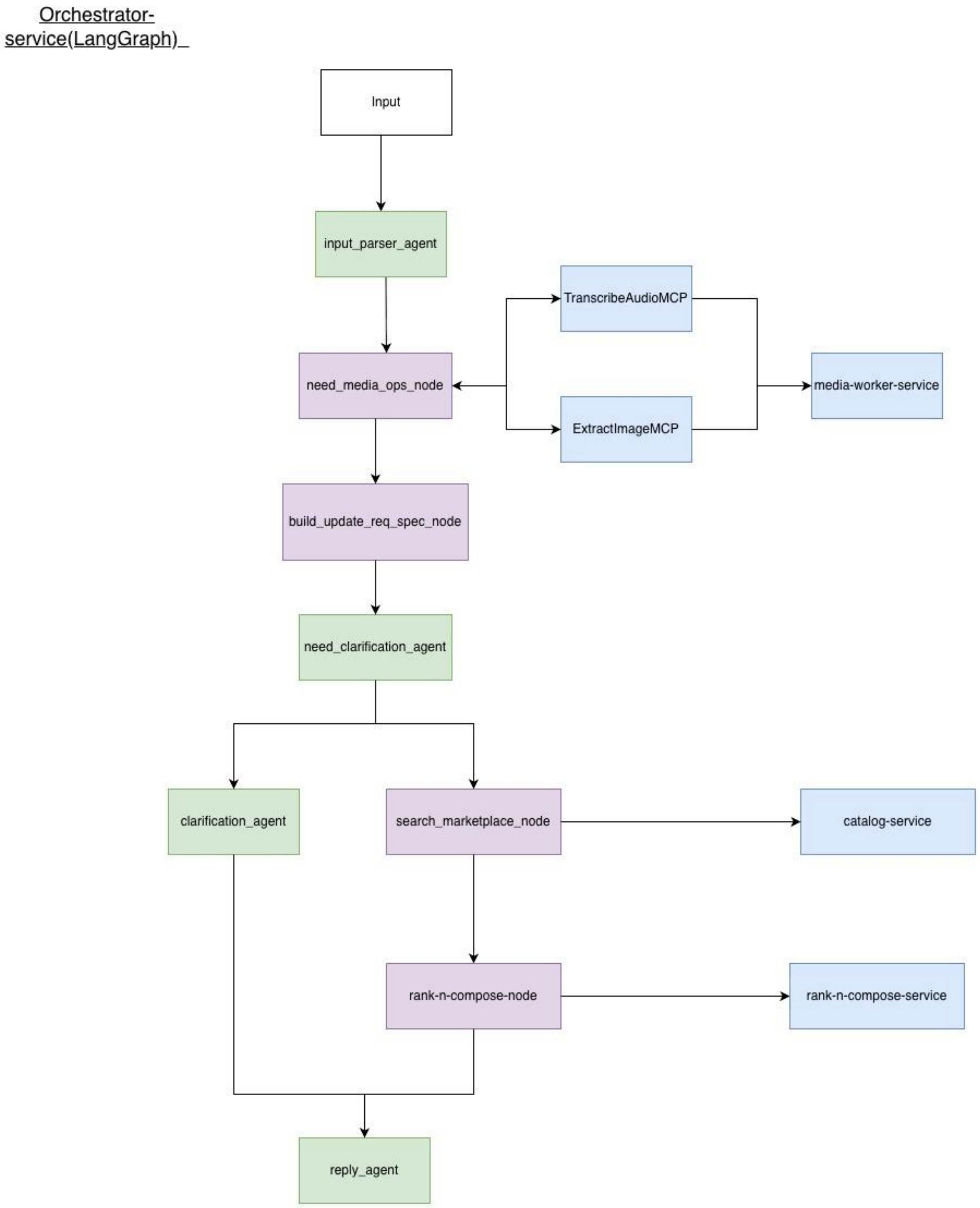
Key Responsibilities:
- Execute product searches across multiple marketplaces in parallel
- Aggregate and normalize results from different marketplace APIs
- Rank results by relevance, price, rating, and other factors
- Return structured product data to the orchestrator

Architecture Components:
- Search Adapters: Marketplace-specific adapters that translate RequirementSpec to marketplace API queries
- Aggregation Engine: Combines results from multiple marketplaces into a unified format
- Ranking Algorithm: Scores and ranks products based on multiple criteria
- Caching Layer: Redis-based caching for frequently searched queries

Supported Marketplaces: eBay, Amazon, Walmart, Best Buy (search operations)

Performance Characteristics: Synchronous operation with 1-3 second response time, designed to handle 100 searches per minute per instance. The service scales horizontally to handle peak loads.

Figure 2.1: Buyer Flow State Machine Graph



2.3 Seller Crosspost Service (Seller Flow)

The Seller Crosspost Service manages asynchronous listing creation across multiple marketplaces.

Technology Stack: FastAPI, Amazon SQS, DynamoDB, Marketplace APIs

Key Responsibilities:

- Validate listing requirements
- Create SQS jobs for each target marketplace
- Track job status in DynamoDB
- Coordinate worker processes that execute marketplace API calls
- Return job confirmation immediately and notify users when complete

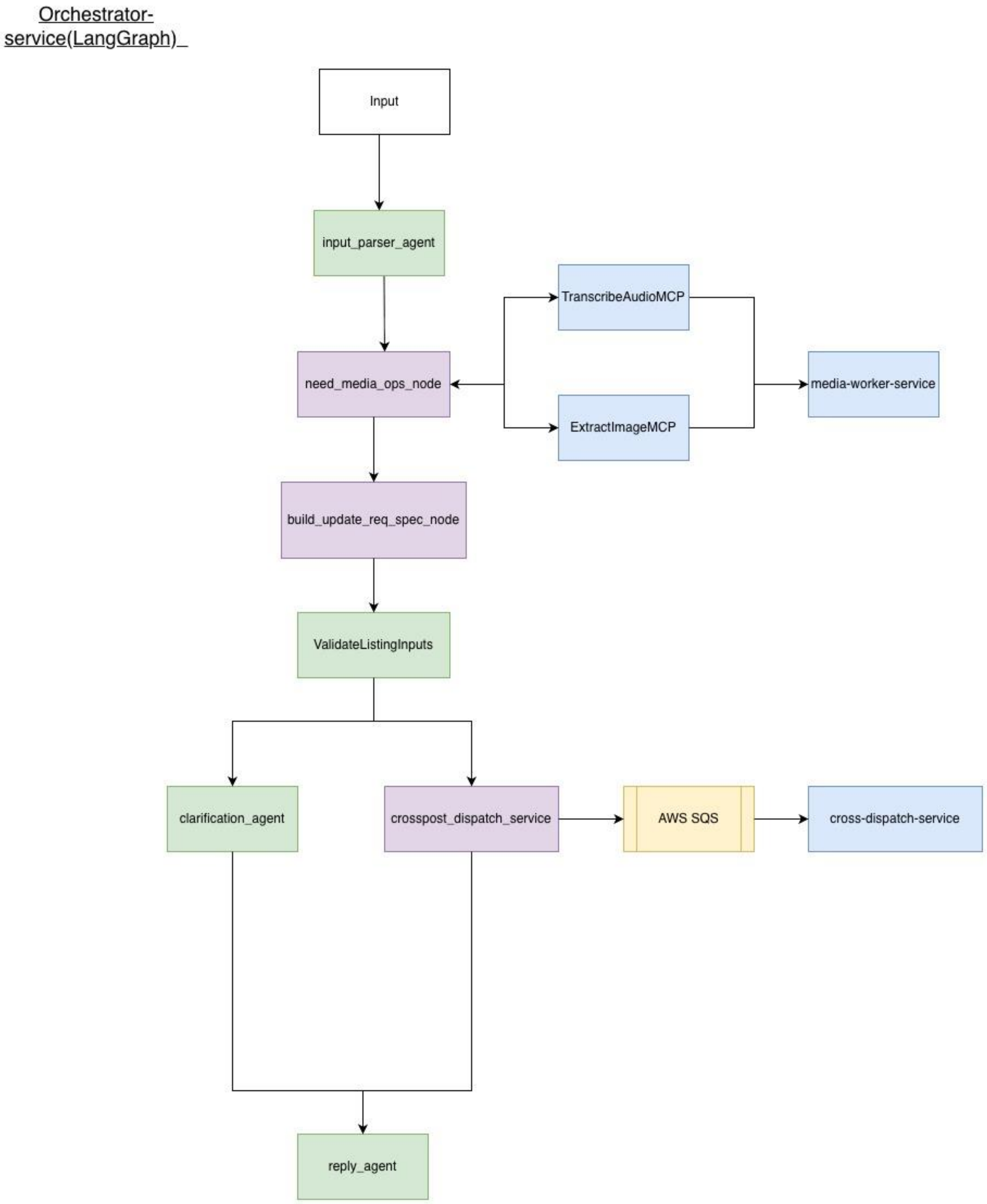
Architecture Components:

- Job Dispatcher: Creates SQS messages for each marketplace posting operation
- SQS Workers: Background workers that consume messages and execute marketplace API calls
- Job Tracker: DynamoDB-based tracking of job status across all marketplaces
- Posting Adapters: Marketplace-specific adapters that format listings according to each platform's requirements

Supported Marketplaces: eBay, Craigslist, Facebook Marketplace, Poshmark (posting operations)

Performance Characteristics: Asynchronous operation with immediate job_id return (100ms), actual posting completes in 30 seconds to 5 minutes depending on marketplace API response times. The service uses SQS for reliable job processing with automatic retries.

Figure 2.2: Seller Flow State Machine Graph



2.4 Media Service

The Media Service processes audio and image inputs for multi-modal interaction.

Technology Stack: FastAPI, AWS Bedrock (for vision and speech), S3

Key Responsibilities:
- Transcribe audio files to text using speech-to-text capabilities
- Extract attributes from images using vision models
- Store media files in S3
- Return processed results to the orchestrator

Architecture Components:
- Audio Transcription Module: Integrates with AWS Bedrock for speech-to-text
- Image Analysis Module: Uses vision-language models to extract product attributes from images
- S3 Storage: Manages media file storage and retrieval

Performance Characteristics: Audio transcription typically takes 2-5 seconds, image analysis takes 1-3 seconds. The service is shared by both buyer and seller flows.

2.5 Client Applications

Client applications provide the user interface for interacting with the TalknShop platform.

Web Application:
- Technology: React, TypeScript, WebSocket, Tailwind CSS
- Features: Chat interface, product card display, media upload, real-time message streaming
- Status: 90% complete with full WebSocket integration

iOS Application:
- Technology: Swift, URLSessionWebSocketTask
- Features: Native iOS interface, voice input, image capture, push notifications
- Status: Planned for future implementation

Both applications connect to the orchestrator service via WebSocket and handle real-time bidirectional communication, including token-by-token LLM response streaming.

2.6 Data Flow and Communication Patterns

The system uses different communication patterns for different operations:

Real-Time Communication: WebSocket connections between clients and the orchestrator service enable real-time bidirectional communication. The orchestrator streams LLM responses token-by-token and sends progress updates during workflow execution.

Synchronous Service Calls: The orchestrator makes HTTP/REST calls to catalog-service and media-service for synchronous operations like product search and media processing. These calls use retry logic with exponential backoff to handle transient failures.

Asynchronous Job Processing: The seller flow uses Amazon SQS for asynchronous job processing. When a user wants to post a listing, the seller service creates SQS messages for each marketplace, returns a job_id immediately, and workers process the jobs in the background.

State Persistence: Conversation state persisted to DynamoDB after each workflow node execution, enabling session resumption and debugging. State includes user messages, extracted requirements, search results, and workflow progress.

Error Handling: The system implements comprehensive error handling with retry logic, graceful degradation (e.g., continuing workflow if media processing fails), and user-friendly error messages sent via WebSocket events.

Table 2.2: Buyer Flow Node Descriptions

Node	**Type**	**Description**
ParseInput	Tool	Load session, normalize message, extract metadata
NeedMediaOps	Agent/LLM	Decide if audio transcription or image processing needed
TranscribeAudio	Tool	Call media-service for speech-to-text

ExtractImageAttrs	Tool	Call media-service for image analysis
BuildRequirementSpec	Agent/LLM	Extract structured requirements from natural language
NeedClarify	Agent/LLM	Assess if requirement spec is sufficient for search
AskClarifyingQ	Agent/LLM	Generate contextual clarifying questions
SearchMarketplaces	Tool	Call catalog-service for product search
RankAndCompose	Tool	Rank results and compose response message
Done	Terminal	Mark workflow complete and return final response

Table 2.3: Seller Flow Node Descriptions

Node	Type	Description
ParseInput	Tool	Load session, normalize message, extract metadata
NeedMediaOps	Agent/LLM	Decide if image processing needed for listing
TranscribeAudio	Tool	Call media-service for speech-to-text (if voice input)
ExtractImageAttrs	Tool	Call media-service for image analysis
BuildListingSpec	Agent/LLM	Extract structured listing from natural language
ValidateListingInputs	Tool	Validate listing requirements per marketplace
NeedClarify	Agent/LLM	Assess if listing spec is sufficient
AskClarifyingQ	Agent/LLM	Generate clarifying questions for missing info
CrosspostDispatch	Tool	Create SQS jobs for each target marketplace
ProcessMarketplaceJobs	Tool	Workers process SQS messages asynchronously
UpdateJobStatus	Tool	Update job status in DynamoDB as jobs complete
ComposeConfirmation	Tool	Generate confirmation message with listing links

Table 2.4: Performance Characteristics by Service

Service	Operation Type	Response Time	Throughput
Orchestrator	WebSocket connection	< 500ms	1000 concurrent
Orchestrator	Workflow execution	2-10s	100 concurrent
Catalog Service	Product search	1-3s	100 searches/min
Seller Service	Job creation	< 100ms	50 jobs/min
Seller Service	Job processing	30s - 5min	Async via SQS

Glossary

LLM (Large Language Model): A type of artificial intelligence model trained on vast amounts of text data to understand and generate human-like text. TalknShop uses Claude 3 Sonnet via AWS Bedrock.

LangGraph: A framework for building state machines with LLMs, enabling complex workflow orchestration with conditional routing and state persistence.

WebSocket: A communication protocol providing full-duplex communication channels over a single TCP connection, used in TalknShop for real-time bidirectional chat.

State Machine: A computational model that defines a set of states and transitions between states based on inputs. TalknShop uses state machines to orchestrate buyer and seller workflows.

RequirementSpec: A structured data model representing product search requirements extracted from natural language, including product type, price filters, and attributes.

ListingSpec: A structured data model representing product listing information for cross-posting to multiple marketplaces, including title, price, condition, and images.

DynamoDB: Amazon's NoSQL database service used in TalknShop for session state persistence and job tracking.

SQS (Simple Queue Service): Amazon's message queuing service used in TalknShop for asynchronous processing of marketplace posting jobs.

AWS Bedrock: Amazon's managed service for accessing foundation models, used in TalknShop for LLM inference with Claude 3 Sonnet.

Microservices Architecture: An architectural approach where applications are built as a collection of loosely coupled, independently deployable services. TalknShop follows this pattern with separate services for orchestrator, catalog, seller, and media operations.

Multi-Modal Input: The ability to process different types of input (text, voice, images) in a unified manner. TalknShop supports text messages, audio transcription, and image analysis.

Cross-Posting: The process of posting a single product listing to multiple marketplaces simultaneously. TalknShop automates this for sellers.

Orchestrator Service: The central coordination service in TalknShop that manages WebSocket connections, executes LangGraph workflows, and coordinates with downstream services.

Catalog Service: The service responsible for product search across multiple marketplaces in the buyer flow.

Seller Crosspost Service: The service responsible for asynchronous listing creation across multiple marketplaces in the seller flow.

References

Anthropic. (2024). Claude 3 Sonnet Model Card. Anthropic AI. Retrieved from https://www.anthropic.com/claude

AWS. (2024). Amazon Bedrock Developer Guide. Amazon Web Services. Retrieved from https://docs.aws.amazon.com/bedrock/

AWS. (2024). Amazon DynamoDB Developer Guide. Amazon Web Services. Retrieved from https://docs.aws.amazon.com/amazondynamodb/

LangChain. (2024). LangGraph Documentation. LangChain AI. Retrieved from https://langchain-ai.github.io/langgraph/

FastAPI. (2024). FastAPI Documentation. FastAPI. Retrieved from https://fastapi.tiangolo.com/

Richardson, L., & Ruby, S. (2007). RESTful Web Services. O'Reilly Media.
Newman, S. (2021). Building Microservices: Designing Fine-Grained Systems (2nd ed.). O'Reilly Media.

Fowler, M. (2014). Microservices. Retrieved from https://martinfowler.com/articles/microservices.html

WebSocket API. (2024). The WebSocket API. MDN Web Docs. Retrieved from https://developer.mozilla.org/en-US/docs/Web/API/WebSocket

React. (2024). React Documentation. Meta. Retrieved from https://react.dev/

TypeScript. (2024). TypeScript Documentation. Microsoft. Retrieved from https://www.typescriptlang.org/docs/

Pydantic. (2024). Pydantic Documentation. Pydantic. Retrieved from https://docs.pydantic.dev/

Appendices

Appendix A: Description of Implementation Repository

The TalknShop project implementation is hosted in a Git repository with the following structure:

Repository Structure:
- `apps/orchestrator-service/` - Central coordination service with WebSocket and LangGraph implementation
- `apps/catalog-service/` - Product search service for buyer flow
- `apps/seller-crosspost-service/` - Asynchronous listing service for seller flow
- `apps/media-service/` - Audio transcription and image processing service
- `apps/talknshop-web/` - React-based web application
- `infrastructure/cdk/` - AWS CDK infrastructure as code
- `tools/documentation/` - Design documents and diagrams

Key Implementation Files:
- Orchestrator Service: `apps/orchestrator-service/main.py`, `app/graph/graph.py`, `app/websocket/manager.py`
- Service Clients: `app/services/catalog_client.py`, `app/services/media_client.py`
- Data Models: `app/models/schemas.py`, `app/models/enums.py`
- Database: `app/db/dynamodb.py`
- Configuration: `app/core/config.py`, `app/core/aws_clients.py`

Technology Stack:
- Backend: Python 3.11+, FastAPI, LangGraph, LangChain
- Frontend: React, TypeScript, Tailwind CSS
- Infrastructure: AWS (Bedrock, DynamoDB, SQS, ECS, S3)
- Development: Docker, Docker Compose

Documentation:
- Architecture documentation: `ARCHITECTURE.md`
- Design specifications: `apps/orchestrator-service/Design-Specification.md`
- Service READMEs: Individual service directories contain detailed README files
- Database documentation: `DATABASE_DOCUMENTATION.md`

Repository Access:
The repository is maintained as part of the SJSU Master's Project and contains comprehensive documentation, code comments, and implementation guides for all services.