TalkNShop: Conversational AI Marketplace

Project Workbook

By

FNU Sameer

Puneet Bajaj

Rutuja Nemane

Kalpesh Patil

11/13/2025

Advisor: Prof. Vidhycharan Bhaskar

Chapter 1. Literature Search, State of the Art

Literature Search

Recent research in conversational AI and multimodal learning has demonstrated significant advancements in creating intelligent and context-aware systems. Foundational studies on large language models (LLMs) such as GPT-3 and GPT-4 have shown strong few-shot reasoning and contextual understanding capabilities across natural language inputs (Brown et al., 2020). Subsequent research has extended these abilities to multimodal contexts, combining text, image, and audio modalities. Models such as CLIP have enabled semantic alignment between text and image for retrieval and recommendation tasks (Radford et al., 2021), while Whisper has achieved robust automatic speech recognition (ASR) across diverse languages and noisy environments (Radford et al., 2022).

The field of conversational recommender systems (CRS) has also grown rapidly. Studies indicate that natural language-based interaction improves personalization, user satisfaction, and trust when compared to traditional recommendation interfaces (Zhang et al., 2018; Jannach & Chen, 2021). CRS models combine preference elicitation with dynamic conversation management, allowing systems to ask clarifying questions before making recommendations.

In parallel, research on multimodal recommender systems has focused on integrating features from different input types—such as textual attributes and product images—to increase recommendation precision (Wei et al., 2019). Similarly, multi-agent LLM frameworks have emerged to improve tool usage, workflow orchestration, and task reliability. LangGraph and similar agent controllers use deterministic execution paths to reduce hallucinations and enforce structured decision-making (Wang et al., 2023).

These academic findings support the design of a unified conversational marketplace capable of interpreting multimodal input, clarifying user intent, and delivering explainable recommendations. The integration of text, audio, and image inputs, combined with agent-based orchestration, represents the current frontier of applied artificial intelligence for commerce and recommendation systems.

State-of-the-Art Summary

Modern conversational and multimodal AI technologies are reshaping the way users interact with digital marketplaces. The state-of-the-art in this field is defined by the convergence of large language models, multimodal processing tools, and cross-platform integration frameworks.

LLM Orchestration and Agents: Industry solutions now use LLMs not only for language generation but also for structured reasoning and orchestration. Tools such as LangChain and LangGraph enable multi-step reasoning through defined agent workflows. These frameworks allow AI systems to perform clarification, search, ranking, and explanation in a structured and explainable way.

Multimodal Input Technologies: Speech-to-text and image understanding models are key enablers of natural, flexible user interaction. Whisper provides reliable voice transcription suitable for real-time applications, while Amazon Rekognition supports visual attribute extraction for “shop by photo” scenarios. Together, these tools enhance accessibility and usability for shoppers who prefer voice or visual queries.

Marketplace Integration and Cloud Infrastructure: State-of-the-art architectures for scalable applications use cloud-native services such as AWS Bedrock, API Gateway, and DynamoDB. These technologies provide modularity, low-latency response times, and support for real-time API integrations with major marketplaces like Amazon, Walmart, and eBay. Asynchronous communication through services such as SQS and EventBridge improves system reliability and fault tolerance.

Industry Benchmarking: Although platforms like Amazon Rufus, Klarna AI, and Shopify Sidekick have implemented conversational assistants, they remain restricted to single ecosystems. In contrast, a unified multimodal conversational marketplace that supports both buying and selling across multiple retail and resale platforms is not yet available. Combining multimodal interaction, cross-marketplace orchestration, and explainability in a single system represents a new step in conversational commerce.

The technologies and frameworks incorporated in the TalkNShop system—LLM orchestration with LangGraph, speech and vision processing with Whisper and Rekognition, and cross-platform data normalization—collectively align with the latest academic research and industry best practices. These components establish a strong foundation for building a next-generation conversational AI marketplace.

References

1. Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. Advances in Neural Information Processing Systems, 33, 1877–1901.

This paper introduced GPT-3, demonstrating that large transformer-based language models can perform reasoning and knowledge tasks with minimal supervision. It forms the academic foundation for conversational and reasoning capabilities used in the TalkNShop AI pipeline.

2. Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., ... & Sutskever, I. (2021). Learning transferable visual models from natural language supervision (CLIP). Proceedings of the 38th International Conference on Machine Learning (ICML).

CLIP aligns visual and textual embeddings to support multimodal search and classification. Its architecture inspires TalkNShop's image-to-product matching and color/feature extraction in the catalog service.

3. Radford, A., Kim, J. W., Xu, T., Brockman, G., McLeavey, C., & Sutskever, I. (2022). Robust speech recognition via large-scale weak supervision (Whisper). arXiv:2212.04356.

Whisper provides accurate speech-to-text transcription across accents and noisy conditions. It is directly relevant to TalkNShop's voice-based search feature that allows users to shop through spoken queries.

4. Zhang, Y., Chen, X., & Zhou, Y. (2018). Towards conversational recommender systems. Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 367–375.

The authors propose one of the first conversational recommendation models integrating natural-language input with ranking logic. This work motivates TalkNShop's approach of using AI-driven clarification and ranking for e-commerce recommendations.

5. Wei, J., Chen, C., & Wang, Y. (2019). A survey on multimodal recommender systems. arXiv:1907.10237.

This survey analyzes multimodal fusion techniques for recommendation, including joint use of text, images, and audio. It directly informs TalkNShop's multimodal design combining image and voice inputs for improved accuracy.

6. Wang, X., Jiang, Y., Liu, Q., & Zhou, Z. (2023). A survey on large language model-based agents. arXiv:2308.11432.

This recent survey discusses emerging frameworks like LangChain and LangGraph that manage multi-agent reasoning and tool execution. It validates the project's use of agent orchestration to ensure reliable, explainable interactions.

Chapter 2. Project Justification

E-commerce and peer-to-peer resale platforms have become essential parts of modern consumer behavior, yet they remain fragmented and inefficient. Buyers often spend significant time switching between multiple sites such as Amazon, Walmart, and eBay to compare options, while sellers must repeatedly enter product details on different resale platforms like Craigslist, OfferUp, and Facebook Marketplace. This duplication of effort, combined with the limitations of keyword-based search, leads to poor personalization and user frustration.

The TalkNShop project addresses these inefficiencies by creating a conversational, multimodal AI marketplace that combines buying and selling within a single unified interface. Users can communicate naturally with the system using text, voice, or images, and the AI agent interprets their intent, asks clarifying questions, and retrieves or posts products accordingly. The use of large language models (LLMs) and multimodal input processing allows TalkNShop to understand natural human communication and translate it into structured product requirements or listings. This reduces manual effort while improving accuracy, speed, and personalization in online shopping and resale.

From a technological standpoint, the project integrates state-of-the-art AI frameworks and cloud infrastructure. It leverages LangGraph for orchestrating multi-agent workflows, Whisper for speech recognition, Amazon Rekognition for image understanding, and OpenAI/AWS Bedrock for reasoning and text generation. These technologies enable explainable and deterministic AI behavior—a key improvement over generic chatbots that operate as black boxes. The architecture is built using AWS services such as API Gateway, Lambda, DynamoDB, and S3 to ensure scalability, security, and high availability.

Academically, the project contributes to the growing research area of multimodal conversational systems and AI-driven commerce automation. It demonstrates practical applications of AI orchestration and multimodal understanding, which are both current research topics in artificial intelligence and human-computer interaction. Professionally, TalkNShop aligns closely with industry trends in AI commerce, recommendation systems, and automation, offering experience in full-stack system design, cloud integration, and applied AI—skills in high demand across the technology sector.

Given its novelty, relevance, and potential impact, TalkNShop represents a strong academic and professional project. It is expected to deliver publishable results or proof-of-concept findings that can attract further development or funding. By bridging cutting-edge AI research with real-world usability, the project showcases how intelligent, multimodal systems can redefine digital shopping and resale experiences.

Chapter 3. Project Requirements

Requirements

The TalkNShop system enables users to buy and sell products through a **conversational, multimodal interface** powered by artificial intelligence. The following requirements describe the functional and non-functional goals of the system based on user needs, academic research, and technical feasibility.

3.1 High-Level Use Cases (Scrum User Stories)

ID	User Story	Acceptance Criteria
US-1	As a buyer, I want to search for products using text, image, or voice so that I can find relevant items without manually browsing multiple sites.	The system accepts text, voice, or image input and returns at least 5 relevant product results within 2 seconds.
US-2	As a buyer, I want the AI to ask clarifying questions when my request is vague, so that recommendations are more accurate.	The AI generates follow-up questions for incomplete or ambiguous input and improves result precision.
US-3	As a buyer, I want to compare prices and ratings across different marketplaces, so that I can make informed purchasing decisions.	Results are normalized and ranked based on price, rating, and delivery time, with clear source labeling.
US-4	As a seller, I want to describe my product once and automatically list it on multiple marketplaces, so that I don't need to re-enter the same information.	The system converts the description into a structured listing and posts it to at least three marketplaces successfully.

US-5	As a user, I want to view explanations for each recommendation, so that I can understand why the product was suggested.	Each product result displays a short rationale highlighting key attributes (e.g., “under $100,” “fast delivery”).

3.2 Functional Requirements

Essential Features

Requirement ID	Description	Verification Method
FR-1	Support for multimodal input: text, voice (via Whisper), and image (via Rekognition).	Functional testing of all three input types.
FR-2	Conversational AI agent that understands user intent and performs clarifications.	Test with predefined conversation flows and intent recognition accuracy.
FR-3	Integration with at least three marketplaces (Amazon, Walmart, eBay) for product search.	API integration testing with live or sandbox endpoints.
FR-4	Seller-side AI that creates listings and cross-posts to multiple resale platforms (eBay, Craigslist, OfferUp).	System integration test validating successful posting and confirmation links.
FR-5	Recommendation and ranking engine that merges and orders results across sources.	Compare system ranking output with predefined test datasets.
FR-6	Explainable recommendations that provide users with rationale and transparency.	UI verification ensuring each product result includes “why recommended” notes.

Desired Features

Requirement ID	Description	Verification Method
DR-1	Advanced personalization using persistent user preferences.	Evaluate recommendation accuracy after multiple sessions.
DR-2	Cross-platform support (Web and Android in addition to iOS).	Functionality testing across devices and browsers.
DR-3	Interactive group shopping or shared cart functionality.	End-to-end test with multiple user sessions in a group flow.
DR-4	Analytics dashboard to visualize usage metrics and user trends.	Verify dashboards show real-time system data from API logs.
DR-5	Voice output for AI responses (text-to-speech).	Validate spoken output matches AI text responses.

Optional Features

Requirement ID	Description	Verification Method
OR-1	Integration with additional marketplaces such as BestBuy or Target.	Test retrieval from new marketplace APIs.
OR-2	Integration with affiliate revenue tracking for monetization.	Validate redirects through affiliate links.
OR-3	Multilingual support for non-English queries and results.	Test translation consistency for key flows.

3.3 Non-Functional Requirements

Category	Requirement	Description
Performance	Latency	System must return search results within 2 seconds (P95 target).
Scalability	Concurrent Users	Must handle 100,000 daily active users in the MVP phase.
Reliability	Uptime	System should maintain 99.5% uptime.
Security	Authentication	All users authenticated via AWS Cognito; API secrets stored in AWS Secrets Manager.
Privacy	Data Protection	Store minimal PII; encrypt all user data in transit (HTTPS) and at rest.
Maintainability	Modularity	Services are containerized for easy updates and scaling on AWS ECS/Fargate.
Usability	Accessibility	Interface supports voice input and multimodal interaction for inclusivity.

Chapter 4. Dependencies and Deliverables

Dependencies

Marketplace API Access

The system depends on access to the official APIs of Amazon, Walmart, and eBay for product search and data retrieval. Any restrictions, revocations, or rate limits from these platforms could directly prevent the catalog service from returning results, effectively halting the buying flow.

Resale Platform Integrations

API availability for resale sites such as eBay, OfferUp, or Craigslist is limited or inconsistent. Successful posting of listings relies on these integrations. If APIs are deprecated or unavailable, the selling flow could fail.

AI Infrastructure and Service Availability

Core features depend on third-party AI providers such as OpenAI and AWS Bedrock for language understanding, Whisper for speech-to-text, and Amazon Rekognition for image processing. Outages, API key issues, or changes in pricing limits could disrupt multimodal functionality.

Cloud Infrastructure (AWS Services)

AWS services—including API Gateway, Lambda, DynamoDB, and S3—form the backbone of the backend system. Downtime or configuration issues in these services would impact all user interactions and system availability.

Data Quality and Schema Consistency

Differences in data formats between marketplaces can cause parsing or normalization errors. Without a unified schema, the AI agents and ranking modules cannot function reliably.

Security and Authentication Services

The system relies on AWS Cognito for user authentication and Secrets Manager for credential management. If these services fail or are misconfigured, user access and API security would be compromised.

Team Collaboration and Development Tools

The project depends on version control (GitHub), issue tracking (Jira), and design collaboration (Figma). Misalignment or tool inaccessibility could delay development and integration tasks across the team.

Deliverables

Mobile Application (iOS MVP)

- A fully functional iOS app that supports text, voice, and image-based queries for both buying and selling workflows.
- Integrated conversational UI and explainable product recommendations.

AI Orchestration Pipeline

- A multi-agent workflow developed using **LangGraph**, capable of managing conversation flow, clarifying user intent, and generating structured JSON for downstream services.

Catalog Integration Service

- A backend microservice that connects to Amazon, Walmart, and eBay APIs.
- Performs data normalization, error handling, and ranking aggregation for search results.

Seller Cross-Posting Service

- Backend service that auto-generates and submits listings to resale marketplaces such as eBay and OfferUp.
- Returns confirmation links to the user through the app.

Explainability and Ranking Engine

- Module that provides transparent, human-readable explanations ("why recommended") for each product in the results list.
- Includes rationale and counterfactual reasoning for AI transparency.

Cloud Backend Deployment

- Fully deployed backend on AWS using **API Gateway**, **Lambda**, **DynamoDB**, **S3**, and **EventBridge**.
- Includes monitoring and logging through **CloudWatch** for observability.

Web Dashboard (Phase 2)

- A lightweight admin dashboard for viewing logs, API health, and analytics.
- Helps in debugging and monitoring live system metrics.

Documentation and Presentation Materials

- Technical documentation covering APIs, architecture, and data flows.
- Presentation slide deck and demo video for project showcase.

Chapter 5. Project Architecture

TalkNShop uses a service-oriented, cloud-native design. The system is layered into client, API gateway, orchestration, domain services (media and catalog), and external integrations. Components communicate over HTTPS/REST with JSON payloads; media files use pre-signed S3 URLs.

Frontend
A React Native/Expo mobile client (iOS/Android; web optional) provides login, a chat interface for buying and selling, and multimodal input (text, audio, image). The app authenticates with AWS Cognito and attaches JWTs to all API calls. Images/audio are uploaded directly to Amazon S3 via pre-signed URLs to avoid proxying large files through the backend.

API tier
All requests go through AWS API Gateway, which handles authentication, routing, and throttling. Requests are forwarded to an internal Application Load Balancer (ALB) that targets containerized backend services. Public traffic is TLS-terminated at API Gateway.

Orchestrator service
A FastAPI–based orchestrator-service implements the conversation workflow (parse → clarify → build/update requirement spec → search → rank/compose → reply). It coordinates calls to downstream services via REST and maintains request state. Deterministic step ordering provides traceability and predictable behavior.

Media path
When input includes audio or images, the orchestrator invokes the media-service.

- Audio is transcribed using AWS Transcribe (ASR), with media retrieved from S3.
- Images are analyzed via Amazon Rekognition to extract attributes (e.g., color, object/category). Results are returned to the orchestrator to enrich the requirement specification.

Catalog path
For product retrieval, the orchestrator calls catalog-MCP (a lightweight controller) which routes to the catalog-service. The catalog-service integrates with Amazon Product API and Walmart API, normalizes heterogeneous fields into a unified product schema, and returns candidates. Error handling covers pagination, timeouts, and marketplace rate limits.

Ranking and response composition
Candidate products flow to rank-n-compose-service, which scores items using price fit, rating, and delivery constraints, then returns a compact response for the client (product cards plus short “why recommended” rationales).

Data and storage

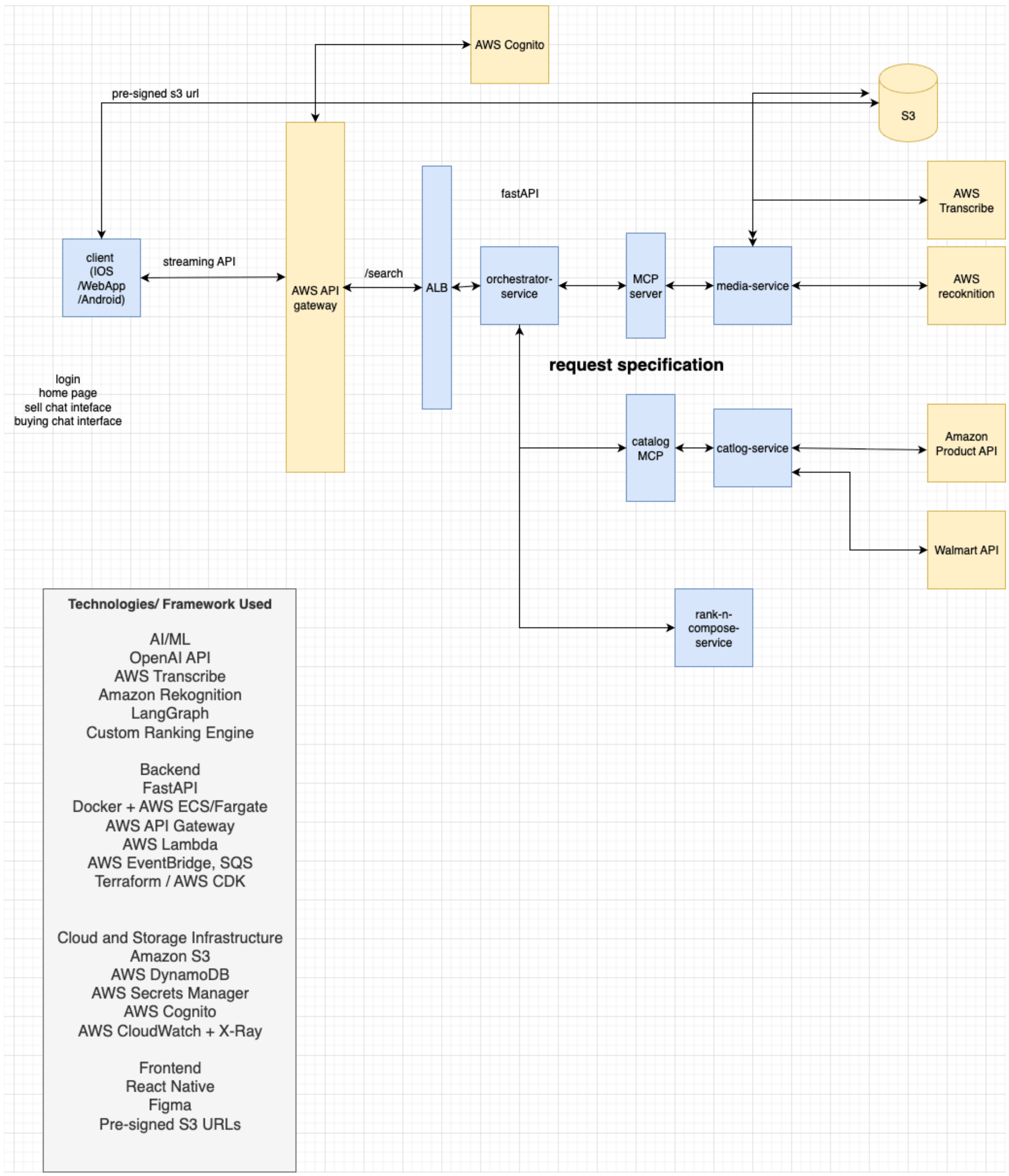
- Amazon S3 stores uploaded media and processed artifacts.
- DynamoDB persists session metadata, requirement specs, and listing/job status.
- Secrets (API keys) are stored in AWS Secrets Manager.
- All services emit logs/metrics to CloudWatch; distributed traces can be captured with AWS X-Ray.

Security
Users authenticate with AWS Cognito; backend services validate JWTs at API Gateway. All data in transit uses HTTPS; S3 objects are encrypted at rest. Pre-signed S3 URLs restrict media upload scope and lifetime.

End-to-end flow

- Client authenticates (Cognito) and uploads media to S3 via pre-signed URL if needed.
- Client sends a /search request to API Gateway → ALB → orchestrator-service.
- Orchestrator enriches input via media-service (Transcribe/Rekognition) when required.
- Orchestrator calls catalog-MCP → catalog-service to query Amazon/Walmart and receive normalized results.
- Orchestrator calls rank-n-compose-service for scoring and rationales, then returns the final response to the client.

Architecture Diagram



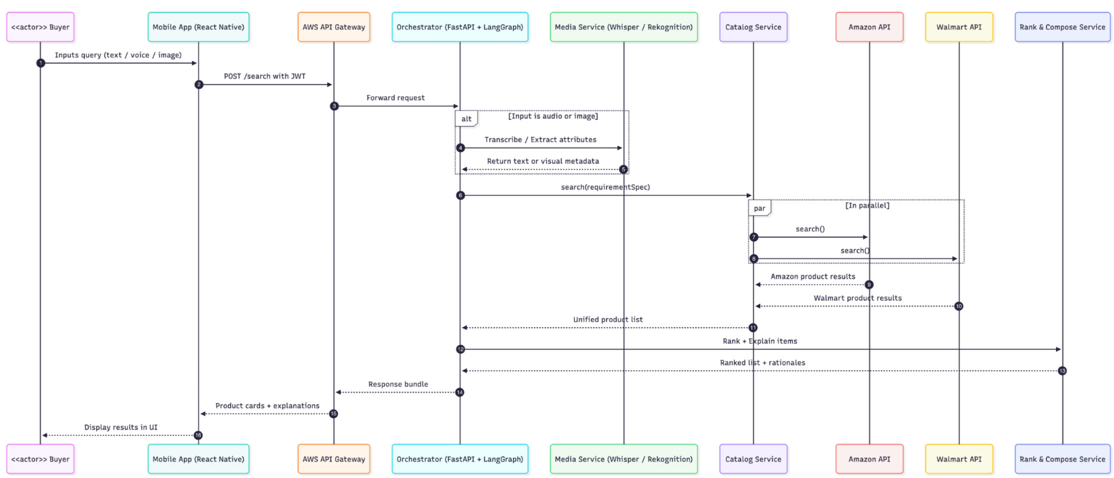
Chapter 6: Project Design

6.1 Overview

The design of the TalkNShop system follows a service-oriented architecture that is modular, scalable, and optimized for AI-powered, multimodal conversational workflows. This chapter includes formal UML diagrams (use case, class, sequence, ER), along with UI mockup placeholders and key design artifacts that together enable a clear understanding of how the system is constructed and how each component contributes to the overall flow. The level of detail provided ensures the project can be successfully completed in the next semester or handed off without loss of continuity.

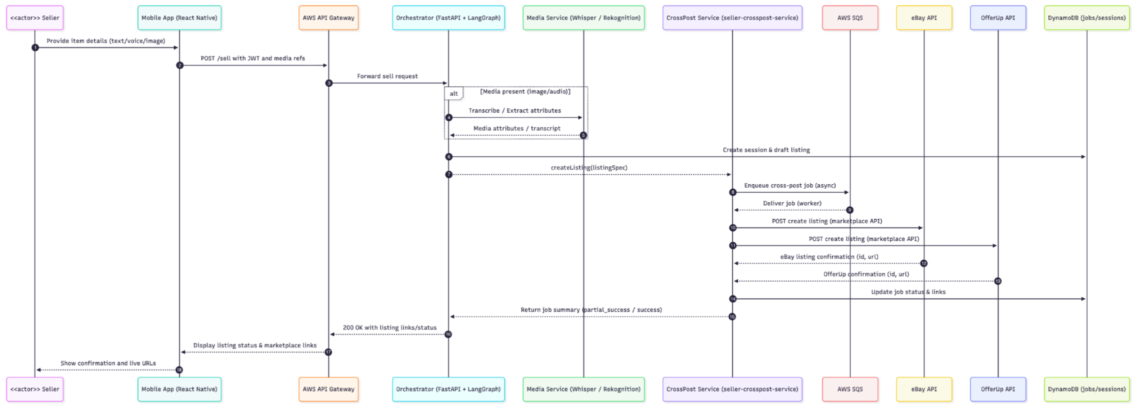
6.2 Sequence Diagrams

6.2.1 Buyer Flow – Product Search



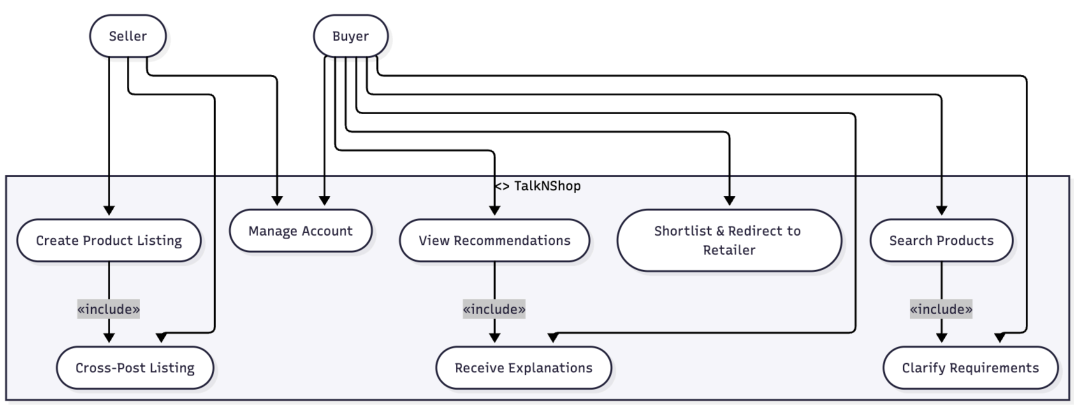
This diagram models the complete end-to-end sequence of a buyer's product search. The user initiates a multimodal query (text, voice, or image), which is routed through API Gateway to the orchestrator. The orchestrator determines whether media processing is needed and, if so, calls the media service (Whisper or Rekognition). It then invokes the catalog service to fetch products from third-party marketplaces and sends the results to the rank-and-compose service, which generates a scored product list with rationales. The response is sent back to the mobile UI.

6.2.2 Seller Flow – Cross-Posting



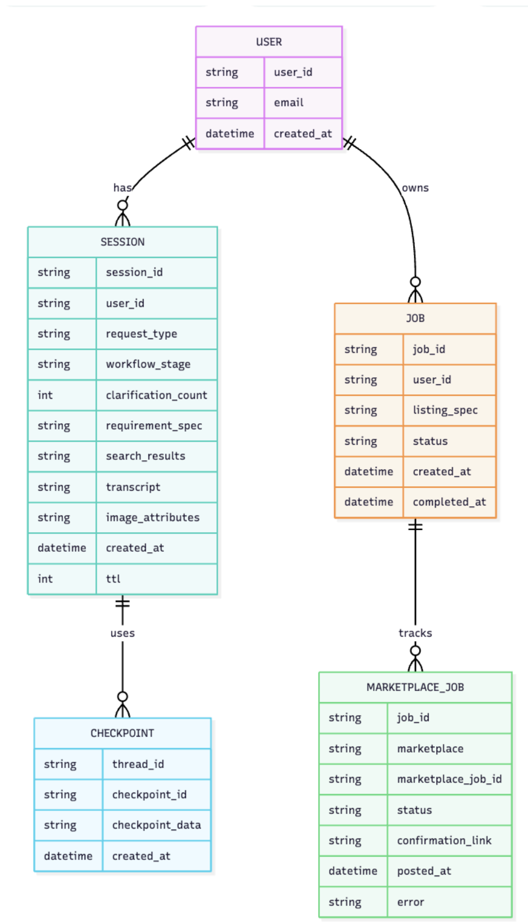
This diagram represents the seller's journey for listing an item. The user inputs details (optionally with voice or image), which the orchestrator processes via the media service. The orchestrator builds a listing spec and hands it off to the seller cross-post service, which enqueues jobs via SQS to post listings to various marketplaces. Confirmation links from each marketplace are returned to the orchestrator and ultimately delivered to the user via the app.

6.3 Use Case Diagram



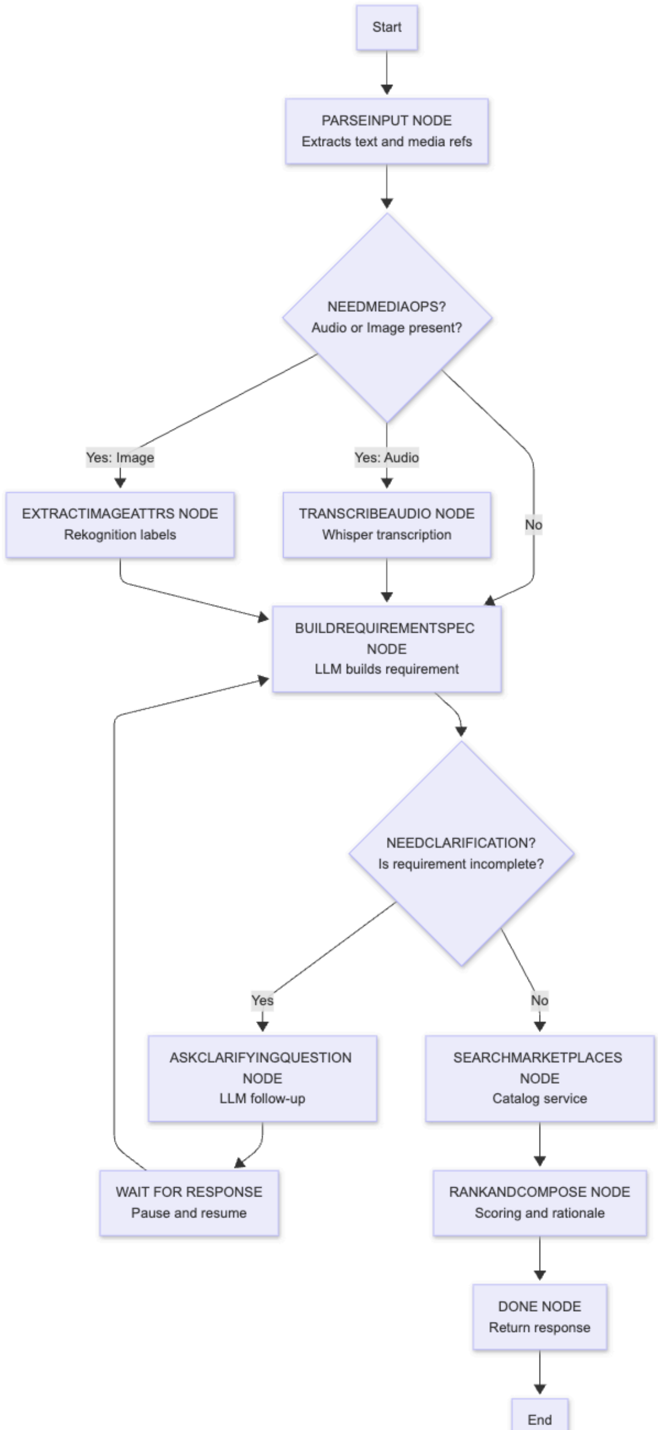
The use case diagram outlines interactions between external actors (Buyer and Seller) and the TalkNShop system. Buyers can search for products using text, voice, or image; receive clarifying questions; and view ranked product recommendations. Sellers can create a listing, which the system will cross-post across multiple marketplaces (e.g., eBay, OfferUp). Both user types can manage their account. The use case model provides a clear high-level map of the system's functional boundaries and supports requirements gathered in Workbook 1.

6.4 Entity Relationship Diagram



This diagram reflects the logical data model used across the three key DynamoDB tables: session tracking, workflow checkpoints, and seller job posting. Each user has multiple sessions and listing jobs. Sessions track conversation metadata (e.g., requirement spec, clarifications, media) and are persisted with a 24-hour TTL. Checkpoints provide workflow resumability via LangGraph. The seller job table stores async task statuses per marketplace. This schema aligns with your production architecture and supports key access patterns documented in your repo.

6.5 Langgraph flow



This diagram illustrates the deterministic flow of buyer queries orchestrated using LangGraph. Based on the input type (text, audio, image), the orchestrator processes the request using AI services like Whisper and Rekognition. Requirements are built and clarified if necessary, followed by marketplace search and ranked product suggestions. The LangGraph-based design supports conditional branching, workflow pausing, and explainability at each stage.

Chapter 7: Project Test Plan

7.1 Test Strategy Overview

TalkNShop follows a **multi-layered test strategy** designed to validate system behavior from the backend services to the frontend UI. Tests are organized into four major categories:

- **Unit Tests** – Test individual service functions (e.g., API clients, orchestrator logic)
- **Integration Tests** – Validate interactions between services (e.g., orchestrator ↔ catalog)
- **System Tests** – End-to-end flows from the mobile app to backend
- **Acceptance Tests** – Validate key use cases with real inputs and expected outputs

The test plan ensures coverage of all functional and non-functional requirements listed in Chapter 3.

7.2 Test Tools and Environment

Tool	Purpose
PyTest	Unit and integration testing (Python services)
Postman	Manual testing of backend APIs
Jest / Detox	React Native frontend testing

AWS CloudWatch	Logs and performance monitoring
Locust	Load and concurrency testing
DynamoDB Local	Local database tests

7.3 Requirement-to-Test Mapping

Requirement	Test Type	Test Description
US-1: Multimodal Search	System + Integration	Submit query via text, image, and voice; validate media processing and search accuracy
US-2: Clarify Requirements	System + Unit	Trigger clarification with vague input; check AI response and retry logic
US-3: Marketplace Comparison	Integration	Query Amazon & Walmart via catalog-service; validate field mapping and normalization

US-4: Seller Listing Flow	System + Integration	Submit listing via app; validate posting across multiple marketplaces via SQS
US-5: Explainable Recommendations	Unit + Integration	Test ranker output includes “why_ranked” string; verify JSON field is passed end-to-end
Non-functional: Latency < 2s	Performance	Simulate load and check P95 response time from orchestrator
Non-functional: Uptime 99.5%	Observability	Monitor service health checks and error rates via CloudWatch
Error handling	Integration + System	Force marketplace API failure; verify fallback and user-facing error message
Security: Auth + Secrets	Unit + System	Validate JWT handling and access control per endpoint
UI validation	Acceptance	Tap-based tests for navigation, loading states, and error boundaries in React Native

7.4 Sample Test Cases

Test Case: Voice Search via Whisper

- **Test Type:** Integration
- **Input:** Audio file saying "I want a red running shoe under 100 dollars"
- **Expected Output:**
    - Audio transcript: matches text intent
    - Requirement spec JSON built correctly
    - MediaService returns attributes
- **Validation:**
    - Logs contain workflow_stage = media_processing
    - Final ranked products match query

Test Case: Seller Cross-Post Job Completion

- **Test Type:** System
- **Steps:**
    - Submit a seller listing with image and text
    - Track SQS job
    - Wait for callback from eBay and OfferUp
- **Expected:**
    - CrossPost service marks job status as - completed
    - Links are stored in DynamoDB
    - App UI displays confirmation screen

7.5 Edge Case Coverage

Scenario	Test Condition
No results from one marketplace	Validate fallback and partial UI
Upload fails during media processing	Verify graceful error message and retry
Invalid user token (expired JWT)	Return 401 error with explanation
Clarification limit exceeded	Auto-complete with fallback search
High load on API	Validate throttling + system stability

7.6 Performance Testing Plan

- Use **Locust** to simulate 100 concurrent users searching products
- Track:
    - **Response time distribution (P95)**
    - **API failures (5xx rate)**
    - **Memory and CPU usage on ECS tasks**
- Success Criteria:
    - No orchestrator crash
    - Product search P95 < 2 seconds
    - API error rate < 1%

7.7 Acceptance Testing Plan

Acceptance testing will be conducted during closed user testing (Sprint 6), using real-world buyer and seller scenarios. At least 15 users will interact with the system and report:

- Search quality
- Response time
- Listing success rate
- Usability and UI satisfaction

A checklist will be used to record whether key flows were successful or failed, and data will be logged for bug triage.

Chapter 8. Implementation Plan and Progress

The TalkNShop project follows an agile, sprint-based implementation plan to ensure steady progress and controlled integration of components. The plan emphasizes setting up the development environment, acquiring essential tools, building foundational prototypes, and iteratively integrating the AI and cloud infrastructure components.

8.1 Programming and Execution Environment

The project is being developed using a cloud-first architecture hosted on Amazon Web Services (AWS). Each team member has individual AWS IAM credentials and sandbox environments for testing.

Key execution environments include:

- AWS ECS/Fargate: Containerized backend microservices including orchestrator, catalog, media, and cross-posting services.
- AWS API Gateway and Lambda: For routing, authentication, and lightweight functions.
- Amazon S3: For media storage (images, audio).
- AWS DynamoDB: For storing structured session data, LangGraph workflow checkpoints, and seller job tracking.
- AWS Bedrock: For LLM operations using Claude 3 Sonnet.

Local Development: Node.js 20, Python 3.11, and FastAPI runtime on macOS and Windows systems.

The mobile client is developed using React Native with Expo, providing a unified execution environment for both iOS and Android. The team uses Expo Go and physical devices for testing, enabling real-time updates during development.

8.2 Development Tools and Frameworks

The team has acquired and configured the following tools and frameworks:

Languages and Frameworks:

- **Frontend**: React Native / Expo, JavaScript, and TypeScript.
- **Backend Services**: Python (FastAPI), LangGraph for orchestration, and boto3 SDK for AWS interactions.
- **AI/ML Stack**: AWS Bedrock/OpenAI API for language models, Whisper for speech-to-text, and Rekognition for image processing.

Cloud and Infrastructure Tools:

- AWS CLI and AWS Console for deployment and monitoring.
- Docker for local container testing.
- Terraform and AWS CDK for infrastructure as code.
- CloudWatch and X-Ray for logging and tracing.

Collaboration Tools:

- **Version Control:** GitHub repository with protected branches.
- **Issue Tracking**: Jira for sprint planning and backlog management.
- **Design and Documentation**: Figma for UI mockups and Lucidchart/Draw.io for architecture diagrams.

8.3 Initial Prototype Development

The first prototype focuses on validating the full pipeline from the mobile app to backend orchestration.

A minimal "prototype of the prototype" has been implemented with the following completed:

1. Hello-world FastAPI service deployed via AWS Lambda and API Gateway for baseline routing.
2. Pre-signed S3 upload tested from the client to validate media-handling flow.
3. Mock marketplace API returning static product results for early UI integration.
4. Chat interface prototype in React Native for basic text conversation with the orchestrator.
5. LangGraph state machine implemented with 10 workflow nodes supporting buyer flow.
6. Real-time orchestration with WebSocket messaging and token streaming enabled.
7. Catalog service integrated with Amazon (via RapidAPI) and Kroger APIs.
8. Media service integrated with Whisper (transcription) and Rekognition (image analysis).

These initial steps ensure that the development environment is functional and all tools are integrated correctly before implementing full orchestration.

8.4 Iterative Implementation Plan (Sprints)

Sprint 1 (Weeks 1–2):
Repository setup, CI/CD pipeline configuration, architecture decision records (ADRs), and schema definitions for products and listings.

Sprint 2 (Weeks 3–4):
Implementation of text-based buyer flow (text → orchestrator → mock catalog service → ranked product cards).

Sprint 3 (Weeks 5–6):
Integration of multimodal input (image and audio) using Rekognition and Whisper.

Sprint 4 (Weeks 7–8):
Seller workflow with structured listing generation and mock posting service.

Sprint 5 (Weeks 9–10): (Currently here)
API integration with real marketplace connectors (Amazon, Walmart, Kroger) and ranking logic implementation.

Sprint 6 (Weeks 11–12):
LangGraph orchestration completed, error handling, metrics, and tracing added. System tested end-to-end with real input. Manual user testing initiated.

Sprint 7 (Week 13):
Demo preparation, documentation finalization, and deployment of MVP to AWS environment. Infrastructure adjustments for observability and WebSocket optimization.

8.5 Progress Tracking and Review

Each sprint concludes with a team demo and review meeting with the advisor. Tasks are tracked through Jira, and commits are reviewed via pull requests on GitHub. Testing is continuous, with unit and integration tests automated through GitHub Actions.

Performance metrics such as API latency, throughput, and error rates are monitored via CloudWatch dashboards. Logs are collected in structured JSON format and reviewed periodically. Manual end-to-end testing is ongoing across buyer and seller flows.

8.6 Implementation Goals

By the end of Semester 1, the goal is to deliver a working iOS MVP that supports:
- Text, image, and audio-based search queries.
- AI-driven clarifications and recommendations.
- Integration with at least two real marketplaces.
- Explainable recommendations and ranking logic.
- Session tracking, LangGraph state persistence, and seller listing initiation with job queuing.

Semester 2 will focus on expanding to Web and Android platforms, adding caching and monitoring, and building a proof-of-concept for monetization. Planned enhancements include complete seller job fulfillment, advanced ranking strategies, error analytics, and affiliate integration.

Chapter 9. Project Schedule

The TalkNShop project follows an **agile development methodology** using two-week sprints. Each sprint delivers a functional milestone and includes design, development, testing, and review activities. The schedule aligns with the Semester 1 timeline (September–December 2025) and identifies unique contributions from each team member. Progress is tracked through Jira, and sprint reviews are conducted with the advisor at the end of every cycle.

9.1 Semester 1 Schedule (Sep – Dec 2025)

Sprint / Week	Milestone / Deliverable	Key Tasks	Team Lead / Members
Sprint 1 (Weeks 1–2)	Project setup & environment configuration	- Create GitHub repo and project structure - Define ADRs (architecture decisions) - Set up AWS IAM users and sandbox environment - Configure CI/CD pipeline - Define JSON schemas for product & listing data	Sameer (Lead), Rutuja, Puneet, Kalpesh
Sprint 2 (Weeks 3–4)	Buyer Flow – Text-based prototype	- Build chat interface in React Native - Develop basic orchestrator service in FastAPI - Mock catalog API integration - Display sample product results in UI	Kalpesh (Lead), Sameer, Rutuja

Sprint 3 (Weeks 5–6)	Multimodal input integration	- Implement audio-to-text using Whisper - Integrate image attribute extraction via Rekognition - Connect media-service with orchestrator - Add file upload via pre-signed S3 URLs	Rutuja (Lead), Sameer, Kalpesh
Sprint 4 (Weeks 7–8)	Seller Flow – Auto-listing framework	- Implement seller flow in orchestrator - Develop crosspost microservice (mock) - Validate structured listing schema - Test end-to-end flow with dummy data	Puneet (Lead), Sameer, Rutuja
Sprint 5 (Weeks 9–10)	Marketplace API integration	- Integrate Amazon and Walmart APIs - Implement rate limiting and retry logic - Enhance ranking and explanation generation - Conduct performance testing	Sameer (Lead), Puneet, Rutuja
Sprint 6 (Weeks 11–12)	System testing and optimization	- Conduct closed user testing (15–20 users) - Improve response latency and reliability	Kalpesh (Lead), Entire Team

		- Monitor logs with CloudWatch - Fix bugs and polish UI	
Sprint 7 (Week 13)	Final MVP and presentation prep	- Deploy final iOS app on test device - Prepare documentation and demo video - Advisor review and dry run	Entire Team

9.2 Semester 2 Schedule (Jan – May 2026)

Sprint 8 (Weeks 1–2)	Android + Web App foundation	- Build Android version via Expo - Port UI to Web (React Native Web) - Ensure API reusability	Kalpesh (Lead), Rutuja
Sprint 9 (Weeks 3–4)	Full Seller Crosspost implementatio n	- Enable posting to eBay and OfferUp - Implement marketplace job tracking - Return confirmation URLs	Puneet (Lead), Sameer
Sprint 10 (Weeks 5–6)	Redis caching + Retry logic	- Add Redis for search result caching - Implement retry logic for external API failures	Rutuja (Lead), Kalpesh
Sprint 11 (Weeks 7–8)	Affiliate model & Analytics	- Integrate affiliate deep links (Amazon) - Capture analytics: CTR, conversions - Add link tracking	Sameer (Lead), Puneet

Sprint 12 (Weeks 9–10)	Performance, monitoring, and alerts	- Add CloudWatch dashboards + alarms - Conduct performance/load testing - Latency optimization	Kalpesh (Lead), Rutuja, Sameer
Sprint 13 (Weeks 11–12)	Final testing and deployment	- Write unit and integration tests - Polish UI for all platforms - Deploy full system to staging/prod	Entire Team
Sprint 14 (Week 13)	Final report + demo presentation	- Complete project report and documentation - Record demo video - Advisor dry run + class demo	Entire Team