

TalkNShop

IOS App, AWS Cognito  
and Seller Workflow Integration

Design Specification

Version 0.1

Kalpesh Patil  
(kalpeshanil.patil@sjsu.edu)

[Version History](#)

[Introduction](#)

[References](#)

[Requirements](#)

[Functional Overview](#)

[Configuration/ External Interfaces](#)

[Debug](#)

[Logging](#)

[Counters](#)

[Implementation](#)

[Testing](#)

[General Approach](#)

[Unit Tests](#)

[Appendix](#)

# TalknShop

## Version History

Version	Notes
1.0	Completed the UI changes for the Landing page and Seller flow. Integrated AWS Cognito for Authentication

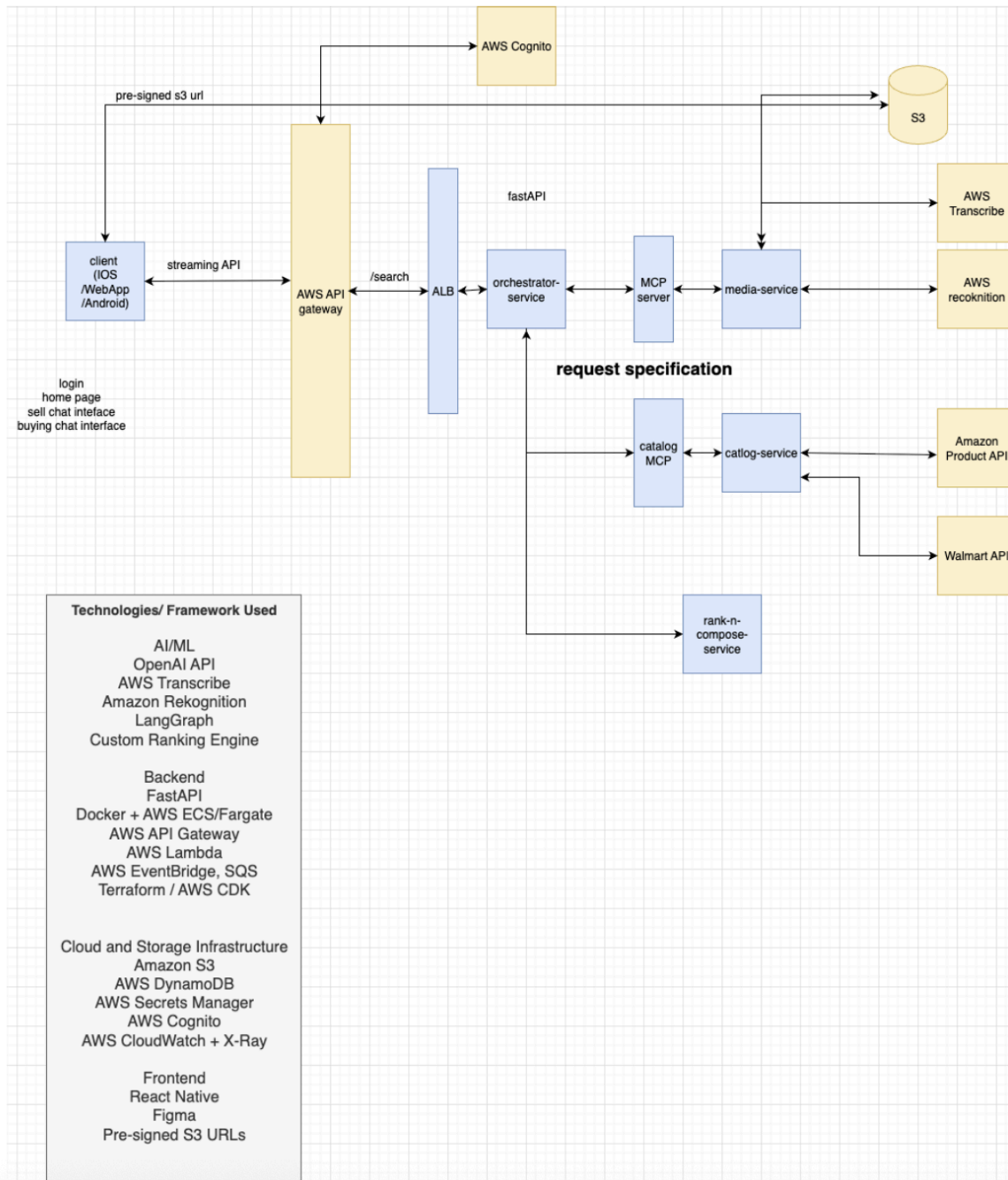
## Introduction

This document explains my individual contribution to our team project. I built the whole mobile UI with React Native and Expo (SDK 54), designed and implemented the UI for the seller flow, and designed and integrated the authentication flow using AWS Cognito (Hosted UI with PKCE). My goal was to make listing a product very fast and keep the sign-in flow simple and secure.

## References

- Team architecture and [workbook notes](#)
- React Native and Expo SDK 54 documentation
- AWS Cognito Hosted UI + PKCE guides
- S3 presigned upload best practices

## Architecture Diagram



System architecture. I own the mobile client and seller workflow.

## Requirements

Item	Detail
Platforms	iOS 15+ and Android 10+
Theme & Accessibility	Dark theme by default, Dynamic Type, good contrast, 44×44pt touch targets
Submit Criteria	Category + at least one Photo
Optional Fields	Name, Brand, Price, Quantity, Description, Condition
Photo Limits	≤5 images; ≤10 MB each; JPG/PNG/HEIC; compress to ~1440px @ ~80% quality
Field Limits	Name ≤120; Brand ≤60; Description ≤2000; Price USD 0–999,999 (2dp); Quantity 1–999
Condition Values	new (default), like-new, good, fair
Authentication	Cognito Hosted UI + PKCE; ID/Access/Refresh tokens; tokens in SecureStore; Access ~1h, Refresh ~30d

## Functional Overview

Tabs: Search, Sell, Chat (placeholder), Wishlist (placeholder), Orders (placeholder), and Profile.

The Search tab features an assistant card, text input, voice, and image entry points, as well as quick suggestions.

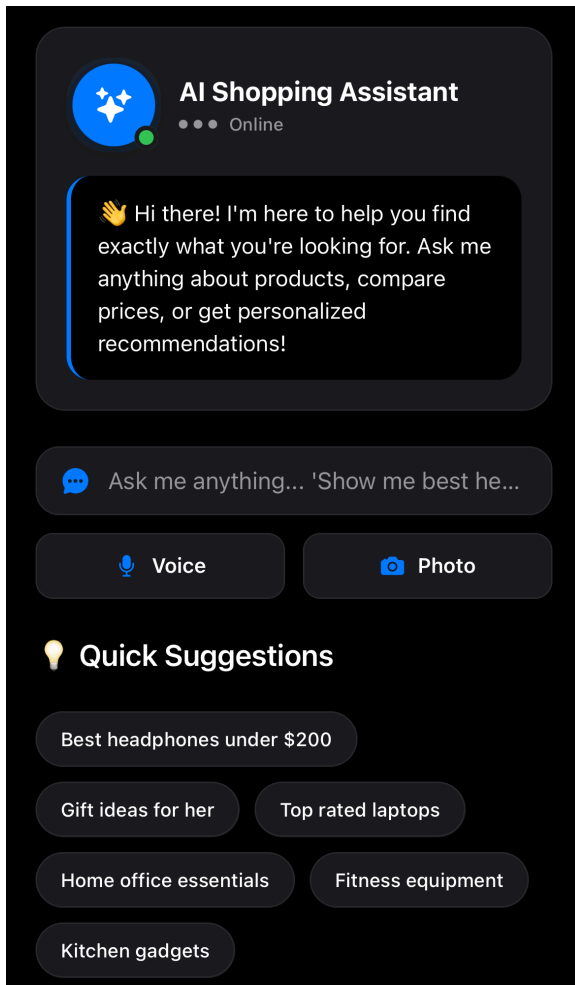
The Sell tab has two steps.

Step 1 is categories and photos.

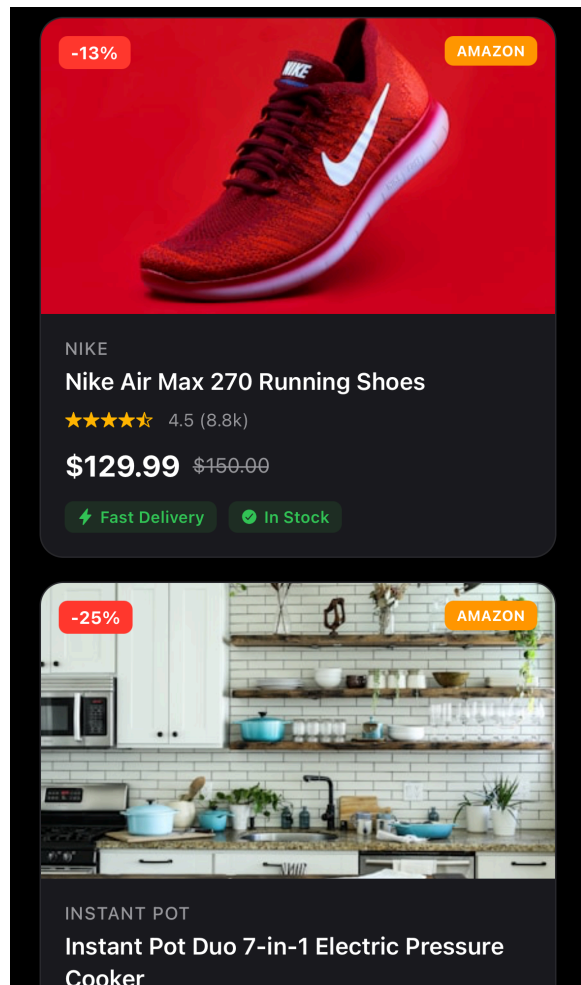
Step 2 is optional details and submit.

On submission, the app uploads images to S3 with presigned URLs and then calls the create listing API.

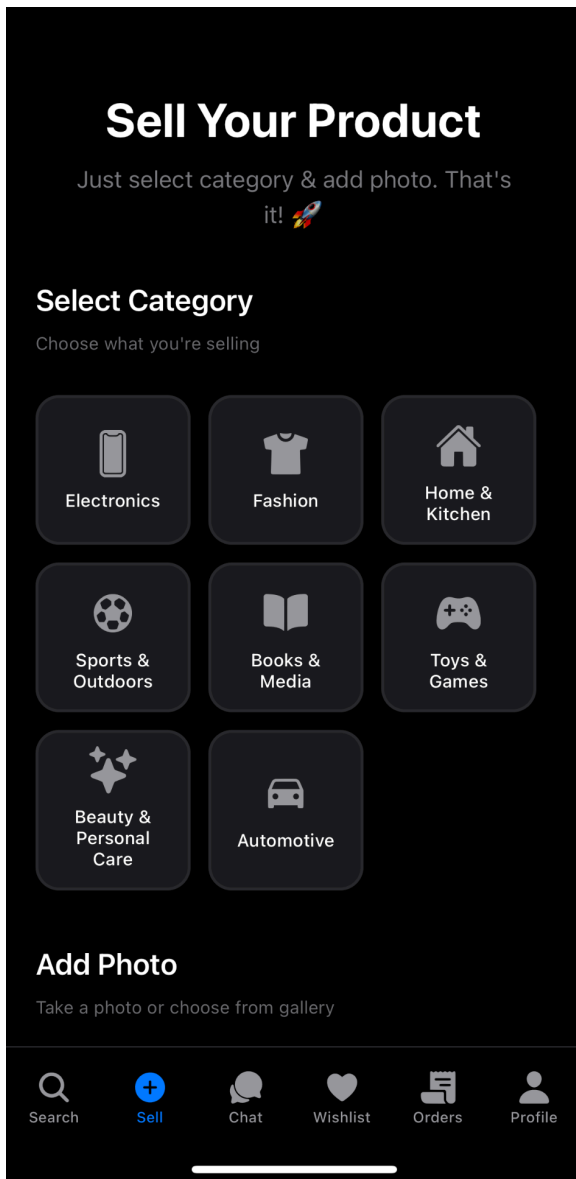
**UI Mockups for the screens:**



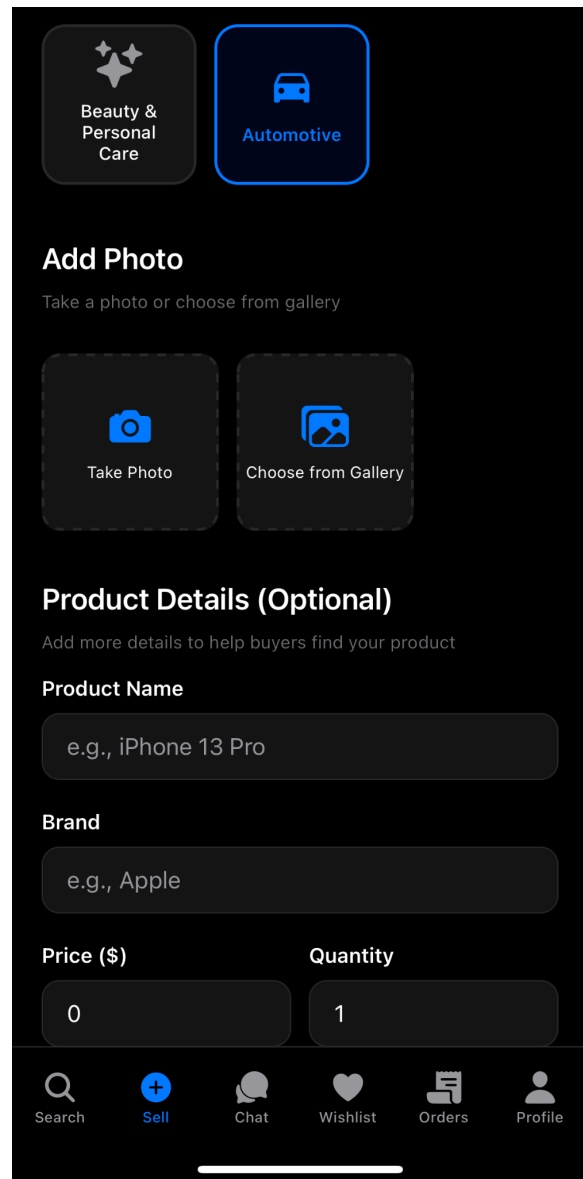
Search using Voice and Photo



Featured products



Sell Screen



Seller Screen Product Details

## Impact & Ownership

Area	Details
Scope I Own	All app UI (tabs, screens, components), full seller listing flow, AWS Cognito login and session handling
Key Outcomes	Cross-platform app, minimal-friction listing (category + photo), secure sign-in with tokens
Backend Touchpoints	Presigned S3 uploads, Create Listing API, Cognito User Pool for accounts
Risk Reduced	Permission handling, upload retry with backoff, token refresh for long tasks

## Effort Breakdown

Week / Hours	Work Done
Week 1	Project setup, navigation, Redux Toolkit, React Query, TypeScript, lint/format
Week 2	Expo SDK 54, React/RN upgrades, theme system, dark mode
Week 3	Search screen, product display, assistant card, suggestions, product modal
Week 4	Seller flow UI, multi-image, optional details, validation, notifications fixes

## Configuration/ External Interfaces

Config	Value
Dev API Base	http://localhost:8000 (v1)
Staging API Base	https://api-staging.talknshop.com (v1)
Prod API Base	https://api.talknshop.com (v1)
Presign Endpoint	POST /media/presign
Create Listing Endpoint	POST /seller/listings
S3 Bucket / Region	talknshop-media / us-east-1; key: listings/{userId}/{uuid}.{ext}
Cognito Domain	talknshop[-env].auth.us-east-1.amazoncognito.com
Redirect Scheme	talknshop://auth (iOS)

Example requests and responses:

```
POST /media/presign
Request:
{
  "fileName": "product-image.jpg",
  "contentType": "image/jpeg",
  "fileSize": 2048576 }
Response:
{
  "uploadUrl": "https://talknshop-media.s3.amazonaws.com/listings/user123/uuid.jpg?... ",
  "key": "listings/user123/uuid.jpg", "expiresIn": 600 }
```

```
POST /seller/listings
Request:
{
  "category": "electronics",
  "name": "iPhone 13 Pro",
  "brand": "Apple",
  "description": "128GB, excellent condition",
  "price": 699.99,
  "quantity": 1,
  "condition": "like-new",
  "images": ["listings/user123/uuid1.jpg"]
}
Success:
{
  "id": "listing-12345",
  "status": "active",
  "createdAt": "2024-11-20T12:00:00Z" }
```

## Debug

The app shows clear messages for missing permissions, large images, and network errors. It saves a draft so the user does not lose work.

## Logging

During development, I log API errors, retries, and upload timing. For analytics I plan events such as `sell_screen_viewed`, `sell_submit_attempted`, `sell_submit_success`, and `sell_submit_error`.

## Counters

Useful counters are the number of photos uploaded, the total submission time, and the retry counts.

## Implementation

Tech stack: Expo SDK 54, React 19, React Native 0.81.5, TypeScript, Expo Router, Redux Toolkit (+persist), React Query, Expo Image Picker/Camera/AV, SecureStore, Reanimated, Gesture Handler, Sentry.  
Sub-tasks: screens and navigation, theme, seller flow UI, draft state, presign and upload, create listing API contract, Cognito sign-in.

# Testing

## General Approach

I test on iOS real devices. I run the happy path and failure paths (no permission, large image, network retry). I also keep a simple end-to-end script that creates a listing.

## Unit Tests

Unit tests cover field validation, draft reducer logic, photo count and size rules, and the upload state machine.

## Appendix

Glossary:

- Presigned URL - a short-lived S3 URL for direct uploads.
- Hosted UI - Cognito sign-in page.
- PKCE - protects the OAuth code flow on mobile apps.