HomeHarbor Instruction Document

Purpose

This document provides a comprehensive, end-to-end view of the HomeHarbor project: the app design, tools used, and the iterative workflow that drives delivery from concept to production.

---

1. App Design (Big Picture)

1.1 Goals

- Build a professional-grade real estate search experience using **real-world data**.
- Demonstrate staff-level engineering practices: **TDD, clean architecture, strong docs, and automation**.
- Enhance the product with **AI-powered assistance** and **vision-based property insights** at **$0 cost** using free-tier services.

1.2 Core Capabilities

- **Property search and filtering** (city, price range, property type, residential type)
- **Sorting and pagination** (price, date, assessed value)
- **Real-world dataset integration** (CT transactions + Redfin market metrics)
- **AI-powered property descriptions** (OpenRouter LLM)
- **Property photo analysis** (Google Street View + OpenRouter vision model)
- **Serverless data pipeline** (Lambda ETL → DynamoDB → S3/CloudFront)

1.3 Data Sources

- **Redfin Data Center**: Market analytics (monthly CSVs, non-commercial use)
- **Connecticut Open Data Portal**: Real Estate Sales 2001–2023 (public domain)
- **Google Street View Static API**: Exterior property photos (free tier)
- **OpenRouter**: Free LLM and vision models (text + image analysis)

1.4 Architecture Overview

Backend (Serverless)

- **AWS Lambda ETL + API**
  - Data ingestion from Redfin + CT Open Data
  - On-demand Street View + AI endpoints
  - Caching with DynamoDB + S3

```
lambda/src/
├── redfin-ingestion.ts
├── ct-socrata-etl.ts
├── street-view-fetch.ts
```

```
├── ai-vision-analysis.ts
└── ai-description-generator.ts
```

Data Storage

```
DynamoDB
├── home-harbor-properties-dev
├── home-harbor-market-metrics-dev
└── home-harbor-ai-insights-dev

S3
├── home-harbor-data-sources-dev
└── home-harbor-images-dev
```

AI Flow (E2E)

1. **Property data requested** → DynamoDB
2. **Street View photo** → S3/CloudFront
3. **Vision model** → Analyze property exterior
4. **LLM** → Generate listing description
5. **Cache results** → DynamoDB (TTL)

Vision Flow (E2E)

1. **Property address → Street View photo**
2. **Photo → Vision model (Molmo 72B)**
3. **AI insight → metadata augmentation** (condition, style, amenities)

---

2. Tools & Technologies

2.1 Core Stack

- **Node.js** (runtime)
- **Jest** (testing)
- **ESLint + Prettier** (lint/format)
- **Playwright** (E2E testing)

2.1.1 AWS Stack (Realtor.com-Ready)

- **AWS Lambda** (ETL + AI analysis functions)
- **Amazon API Gateway** (HTTP API + routing)
- **Amazon DynamoDB** (properties, market metrics, AI caches)
- **Amazon S3** (raw datasets + image cache)
- **Amazon CloudFront** (CDN for property photos)
- **Amazon EventBridge** (scheduled ingestion)
- **Amazon CloudWatch** (logs, metrics, alarms)
- **AWS IAM** (least-privilege access control)
- **AWS Secrets Manager** (API keys for OpenRouter + Google Maps)

2.2 AI & Data

- **OpenRouter API** (LLM orchestration)
- **AllenAI Molmo 72B** (vision model)
- **Google Street View Static API** (property photos)
- **Connecticut Open Data (Socrata API)** (real estate transactions)
- **Redfin Data Center (CSV)** (market analytics)

2.3 Engineering Practices

- **TDD**: RED → GREEN → REFACTOR
- **Conventional commits**
- **Pre-commit security hooks**
- **Architecture Decision Records**

---

3. Iterative Workflow (E2E Project Execution)

3.1 Phase 0 — Foundation

- Configure repository structure
- Add linting, formatting, tests
- Establish architectural guidelines

3.2 Phase 1 — Data Pipeline (Serverless)

- Implement Redfin CSV ingestion (Lambda)
- Implement CT Socrata ETL (Lambda)
- Store properties + metrics in DynamoDB
- Cache images in S3 + CloudFront

3.3 Phase 2 — AI Architecture & Design

- Document OpenRouter model strategy
- Design vision pipeline (Street View + Molmo 72B)
- Add Secrets Manager configuration

3.4 Phase 3 — AI Implementation (TDD)

- Vision analysis Lambda (Molmo 72B)
- Description generator Lambda (Llama 3.3 70B)
- DynamoDB caching + TTL

3.5 Phase 4 — UI & Delivery

- UI search experience
- Property cards with AI insights
- API Gateway integration

3.6 Phase 5 — Deployment

3.6 Phase 5 — Deployment (AWS)

- **Lambda + API Gateway**: `/properties`, `/search`, `/analyze`, `/describe`
- **DynamoDB**: properties, market metrics, AI analyses (TTL)
- **S3**: raw datasets + Street View images
- **CloudFront**: CDN for property photos
- **CloudWatch**: error alarms, latency, and cost metrics
- **IAM + Secrets Manager**: secure key storage and rotation

---

4. E2E Project Narrative

Idea → Real Data → Core Features → AI Enhancements → UI → Deployment

1. **Start with real data** to prove real-world relevance.
2. **Build core search capabilities** with strict TDD.
3. **Design AI integration** to add strategic differentiation.
4. **Implement AI features** with free-tier services to stay cost-effective.
5. **Polish UI and deploy** for recruiter-grade presentation.

---

5. References (Design Docs)

- OpenRouter LLM architecture: docs/OPENROUTER_LLM_ARCHITECTURE.md
- AI feature overview: docs/AI_FEATURES_OVERVIEW.md
- Lessons learned: LESSONS_LEARNED.md

---

6. What’s Next

- Create API Gateway endpoints
- Build React UI for search + AI insights
- Add map-based browsing and filters

Last Updated: January 31, 2026