Product Requirements & Specification Document

Project Name

AgentX - Multi-Agent Collaboration for Real Estate

Description

AgentX is a real estate application showcasing multi-agent collaboration using LangGraph and CrewAI. Agents autonomously negotiate, analyze property listings, and generate tailored property recommendations for users. The application leverages generative AI, Python, LangChain, and OpenAI technologies, with a futuristic, startup-inspired theme.

---

1. Goals & Objectives

Goal	Description
Demonstrate multi-agent collaboration	Showcase negotiation, analysis, and recommendation workflows
Provide property recommendations	Deliver relevant, AI-generated property suggestions
Integrate LangGraph & CrewAI	Utilize both frameworks for agent orchestration
Futuristic, startup-inspired experience	Modern UI/UX and branding

---

2. Core Features

Feature	Description
Multi-Agent Negotiation	Agents discuss and negotiate property options based on user preferences
Listing Analysis	Agents analyze property data (price, location, features)
Recommendation Generation	Agents collaboratively generate and present property recommendations
User Input Interface	Simple UI for users to input preferences and view recommendations
Agent Conversation Visualization	Display agent interactions and reasoning steps (optional, for demo)

---

3. User Stories

As a…	I want to…	So that…
Homebuyer	Input my property preferences	I receive tailored property recommendations
Stakeholder	Observe agent collaboration and negotiation	I can assess the effectiveness of the system
Developer	Extend or modify agent behaviors	I can adapt the system for new scenarios

---

4. Functional Requirements

ID	Requirement
FR1	System accepts user property preferences (location, budget, features)
FR2	Agents analyze a set of property listings (mock or real data)
FR3	Agents negotiate and discuss optimal listings based on user input
FR4	System generates and displays top property recommendations
FR5	Agent interactions are logged and optionally visualized
FR6	Application uses LangGraph and CrewAI for agent orchestration
FR7	UI presents a modern, futuristic, startup-inspired design

---

5. Non-Functional Requirements

ID	Requirement
NFR1	Responsive and performant UI
NFR2	Modular, extensible codebase (Python)
NFR3	Secure handling of user input
NFR4	Clear documentation for setup and extension

---

6. Technical Specifications

Area	Specification
Language	Python 3.9+
Frameworks	LangChain, LangGraph, CrewAI, OpenAI API
Data	Property listings (mock JSON/CSV or API)
UI	Web-based (Streamlit, Flask, or similar lightweight framework)
Deployment	Local or cloud (Docker support recommended)
Visualization	Optional: Agent conversation flow (graph or chat-style)

---

7. Architecture Overview

```
flowchart TD
    UserInput[User Input]
    UI[Web UI]
    Orchestrator[LangGraph + CrewAI Orchestrator]
    Agents[AI Agents (Negotiator, Analyst, Recommender)]
    Listings[Property Listings Data]
    Recommendations[Recommendations Output]
```

```
    UserInput --> UI
    UI --> Orchestrator
    Orchestrator --> Agents
    Agents --> Listings
    Agents --> Orchestrator
    Orchestrator --> Recommendations
    Recommendations --> UI
```

---

8. Sample Agent Interaction (Pseudocode)

```
# Pseudocode for agent collaboration
user_prefs = get_user_preferences()
listings = load_property_listings()

negotiator = NegotiatorAgent()
analyst = AnalystAgent()
recommender = RecommenderAgent()

negotiated_listings = negotiator.negotiate(user_prefs, listings)
analyzed_listings = analyst.analyze(negotiated_listings)
recommendations = recommender.recommend(analyzed_listings, user_prefs)

display_recommendations(recommendations)
```

---

9. Milestones & Timeline

Milestone	Target Date
Requirements & Design Complete	Week 1
Core Agent Logic Implemented	Week 2
UI & Integration	Week 3
Testing & Refinement	Week 4
Demo & Documentation	Week 5

---

10. Success Criteria

- Agents collaborate and generate relevant property recommendations
- User can input preferences and receive results via UI
- Agent interactions are observable (log or visualization)
- System is stable, extensible, and well-documented

---

11. Risks & Mitigations

Risk	Mitigation
API limits/costs	Use mock data or rate-limiting
Agent logic complexity	Start with simple behaviors, iterate
UI/UX delays	Use rapid prototyping frameworks

---

12. Appendix

- **Technologies:** Python, LangChain, LangGraph, CrewAI, OpenAI API
- **Design Theme:** Futuristic, startup-inspired
- **Data:** Use mock property listings for demo purposes
