Dreamhouse Agent Project: Technical Design

1. Introduction

This document outlines the technical design for the Dreamhouse Agent project. It provides detailed specifications for the development team to implement the functionalities described in the Project Kickoff Document.

2. System Architecture

The Dreamhouse Agent will be built on the Salesforce Platform, leveraging Agentforce, Data Cloud RAG, and external APIs. The architecture comprises the following key components:

Salesforce Agentforce:

- Agent definition and dialog management
- Orchestration of agent workflows

Retrieval Augmented Generation (RAG) Engine:

- Leverage Salesforce Data Cloud and Einstein Data Libraries for RAG functionality with PDF files
- RAG will be processing the file titled "Dreamhouse Property Listing Guidelines & General Terms"

API Integration Layer:

- Connectivity to external APIs (walk score)

Internal User Interface:

- Salesforce Slack App

External User Interface:

- Salesforce Experience Cloud

3. Data Model

3.1. Salesforce Objects

- `Property__c`: Stores property details (address, price, features, etc.)
- `Broker__c`: Stores realtor information
- `Contact`: Stores client information (name, contact details, preferences, etc.)
- `Opportunity`: Tracks sales opportunities
- `Case`: Manages client service cases

3.2. External Data

- Property maps: https://leafletjs.com
- Walk Score API: https://www.walkscore.com
- Neighborhood information (schools, amenities, walk scores)

4. Component Design

4.1. RAG Engine

Process:

- Ingest PDF files into Data Cloud
- Configure Einstein Data Libraries to process and index the PDF data for RAG
- Use Prompt Templates to retrieve relevant information based on user queries

Technology:

- Salesforce Data Cloud, Einstein Data Libraries

4.2. API Integration Layer

Component:

- Apex Classes

Functionality:

- Encapsulate API calls to external services
- Handle authentication, request formatting, and response parsing
- Implement robust error handling and rate limiting

Security:

- Use Salesforce Named Credentials for secure storage of API keys

4.3. Agent Development

Component:

- Salesforce Agentforce

Functionality:

- Define agent personas and dialog flows
- Integrate with the RAG Engine and API Integration Layer
- Handle user input and generate responses

Details:

- *Agent Configuration*: Define agent names, descriptions, and initial greetings
- *Dialog Flow Design*: Create dialog flows for each use case (e.g., property search, tour scheduling). Include comprehensive error handling and fallback mechanisms in the dialog flows
- *Handoff to Human Agents*: Configure Omni-Channel to route conversations to appropriate agents. Ensure context is passed to the human agent to provide a seamless user experience
- *Rich Media*: Incorporate images, videos, and interactive elements in agent responses to enhance user engagement
- *Sentiment Analysis*: Use Salesforce Einstein Language or a similar service to analyze user sentiment and adjust agent responses accordingly

Technology:

- Salesforce Agentforce, Experience Cloud, Omni-Channel, Salesforce Flows

4.4. Internal User Interface

Component:

- Salesforce Slack App

Functionality:

- Enable users to interact with the agent from Slack
- Display property information, client details, and notifications
- Support user input and agent responses

Technology:

- Salesforce Slack App, Slack API, Salesforce Flows

4.5. External User Interface

Component:

- Salesforce Experience Cloud

Functionality:

- Provide a portal for property search, tour scheduling
- Embed the agent for chat-based interactions
- Display property details, tour availability, and status updates

Technology:

- Salesforce Experience Cloud, Lightning Web Components

5. Security

Data Access Control:

- Use Salesforce sharing settings and field-level security to control access to data

API Security:

- Use Salesforce Named Credentials to securely store API keys
- Implement authentication and authorization for external APIs
- Implement rate limiting to prevent API abuse

Data Privacy:

- Comply with data privacy regulations (GDPR, CCPA)
- Implement data masking for sensitive information
- Implement audit logging

6. Scalability and Performance

Asynchronous Processing:

- Use Salesforce Queues and Platform Events to handle long-running processes

Code Optimization:

- Follow Salesforce best practices for Apex and Flow development
- Optimize SOQL queries and avoid governor limits

Caching:

- Implement caching mechanisms to improve performance

7. Error Handling

Apex and Flow Error Handling:

- Implement try-catch blocks in Apex to handle exceptions

- Use fault paths in Flows to handle errors
- Log errors to Salesforce logs

API Error Handling:

- Handle API errors gracefully
- Implement retry mechanisms for transient errors
- Provide informative error messages to the user

RAG Error Handling:

- Handle cases where the RAG engine fails to retrieve relevant information
- Provide fallback responses to the user

8. Testing

- Develop unit tests for Apex classes and Flows
- Develop tests for Agentforce agents
- Create integration tests for API integrations and component interactions
- Perform user acceptance testing (UAT) with stakeholders