AI Property Recommendation System - Backend Architecture

System Overview

This document outlines the production-ready architecture for an AI-powered property recommendation system, designed to scale on Azure cloud infrastructure.

System Architecture Diagram

```
┌──────────────────┐    ┌──────────────────┐
┌──────────────────┐
│  Web Frontend  │  │ Mobile Apps  │  │ Partner APIs  │
│  (React/Next)  │   │ (React Native) │  │          │
└─────────┬────────┘    └─────────┬────────┘
└─────────┬────────┘
    │          │          │
    └──────────────────────────┼──────────────────────────────┘
              │
        ┌──────────────────┐
        │  API Gateway   │
        │ (Azure API Mgmt)│
        └─────────┬────────┘
             │
    ┌────────────────────┼────────────────────┐
    │        │        │
  ┌───────▼───────┐  ┌───────▼───────┐  ┌───────▼───────┐
  │ User Service│  │Recommendation│  │ Data Service│
  │(App Service)│  │  Service   │  │(App Service)│
  │       │  │(App Service) │  │
  └───────────────┘  └───────┬───────┘  └───────────────┘
             │
     ┌───────────────┼───────────────┐
     │        │        │
   ┌──────────▼──────────┐  │  ┌──────────▼──────────┐
   │  ML Pipeline   │  │  │ Search Engine  │
   │(Azure ML/Databricks)  │  │(Azure AI Search)  │
   └─────────────────────┘  │  └─────────────────────┘
            │
       ┌──────────▼──────────┐
       │  AI Agent Hub   │
       │ (Azure OpenAI +  │
       │  Custom Agents)  │
       └──────────┬──────────┘
            │

  ┌─────────────────────────────┼─────────────────────────────┐
  │        │        │
  ┌───▼───┐  ┌─────▼─────┐  ┌─────▼─────┐
  │Budget │    │Location │    │Features │
  │Agent  │    │ Agent  │    │ Agent  │
  └───────┘    └───────────┘  └───────────┘
            │
     ┌───────────────────┼───────────────────┐
     │       │       │
  ┌─────────▼─────────┐  ┌─────────▼─────────┐
```

```
┌──────────────▼──────────────┐
│  Cosmos DB   ││  Blob Storage ││ Event Hubs   │
│ (Properties)  ││  (Images/Docs) ││ (Analytics)  │
└─────────────────┘  └─────────────────┘
└──────────────┘
```

Azure Services Mapping

Core Infrastructure

- **Azure App Service**: Host FastAPI backend services
- **Azure API Management**: Gateway, rate limiting, authentication
- **Azure Front Door**: CDN and load balancing
- **Azure Container Registry**: Docker image storage

Data Layer

- **Azure Cosmos DB**: Primary property database (NoSQL)
- **Azure SQL Database**: User profiles and transaction data
- **Azure Blob Storage**: Property images, documents, ML models
- **Azure Data Lake**: Historical data and analytics

AI/ML Layer

- **Azure OpenAI Service**: GPT-4 for explanations and agent coordination
- **Azure Machine Learning**: Custom property scoring models
- **Azure AI Search**: Semantic search and filtering
- **Azure Databricks**: Data processing and model training

Monitoring & Security

- **Azure Application Insights**: Performance monitoring
- **Azure Key Vault**: Secrets and API key management
- **Azure Active Directory B2C**: User authentication
- **Azure Security Center**: Threat detection

API Contracts

1. Property Recommendation Endpoint

POST `/api/v1/recommendations`

Request Schema:

```json
{
  "user_preferences": {
    "budget_min": 200000,
    "budget_max": 1000000,
    "city": "Denver",
    "min_bedrooms": 2,
    "max_commute_time_minutes": 45,
    "min_school_rating": 7,
    "must_have_features": ["pool", "garage"],
    "lifestyle_preferences": ["modern", "family-friendly"],
    "size_preference": "medium"
  },
  "user_context": {
    "user_id": "user_12345",
    "session_id": "session_67890",
    "previous_searches": []
  },
  "options": {
    "max_results": 3,
    "include_explanations": true,
    "enable_ai_insights": true
  }
}
```

Response Schema:

```json
```

```json
{
  "recommendations": [
    {
      "property_id": "prop_98765",
      "address": "123 Example St, Denver, CO",
      "price": 750000,
      "match_score": 92,
      "confidence": 0.87,
      "basic_info": {
        "bedrooms": 3,
        "bathrooms": 2.5,
        "size_sqft": 2200,
        "year_built": 2018,
        "lot_size_sqft": 8000
      },
      "location_info": {
        "city": "Denver",
        "neighborhood": "Capitol Hill",
        "school_rating": 8,
        "commute_time_minutes": 22,
        "walkability_score": 78
      },
      "features": {
        "has_pool": true,
        "garage_spaces": 2,
        "amenities": ["hardwood_floors", "updated_kitchen", "fireplace"]
      },
      "ai_explanation": {
        "summary": "Perfect match for your family with excellent schools and modern amenities",
        "key_reasons": [
          "Within budget with great value at $750k",
          "22-minute commute beats your 45-minute requirement",
          "School rating of 8/10 exceeds your minimum of 7",
          "Modern 2018 construction with desired pool"
        ],
        "potential_concerns": [
          "Slightly smaller lot than neighborhood average"
        ]
      },
      "images": [
        "https://storage.blob.core.windows.net/properties/prop_98765/main.jpg"
      ],
      "agent_insights": {
        "budget_agent_score": 85,
        "location_agent_score": 95,
        "features_agent_score": 90
      }
    }
  ],
  "search_metadata": {
    "total_properties_analyzed": 1247,
    "filters_applied": 8,
    "processing_time_ms": 340,
    "model_version": "v2.1.3"
  },
  "ai_summary": "Based on your preferences, I found 3 excellent properties that balance your budget, location, an
}
```

2. Property Details Endpoint

GET `/api/v1/properties/{property_id}`

Response Schema:

```json
```

```
{
  "property": {
    "property_id": "prop_98765",
    "detailed_info": {
      "full_address": "123 Example St, Denver, CO 80205",
      "mls_number": "MLS123456",
      "listing_status": "active",
      "days_on_market": 15,
      "price_history": [
        {"date": "2024-01-15", "price": 775000, "event": "price_reduction"},
        {"date": "2024-01-01", "price": 800000, "event": "listed"}
      ]
    },
    "comprehensive_features": {
      "interior": {
        "flooring": ["hardwood", "tile"],
        "kitchen": "updated_2022",
        "appliances": "stainless_steel",
        "heating": "forced_air_gas",
        "cooling": "central_air"
      },
      "exterior": {
        "roof": "composite_shingle_2020",
        "siding": "brick_vinyl",
        "landscaping": "professional",
        "outdoor_features": ["pool", "deck", "garden"]
      }
    },
    "neighborhood_data": {
      "demographics": {
        "median_income": 75000,
        "avg_age": 35,
        "family_percentage": 68
      },
      "nearby_amenities": [
        {"type": "grocery", "name": "Whole Foods", "distance_miles": 0.8},
        {"type": "park", "name": "City Park", "distance_miles": 1.2},
        {"type": "hospital", "name": "Denver Health", "distance_miles": 3.1}
      ]
    }
  }
}
```

3. User Feedback Endpoint

POST `/api/v1/feedback`

Request Schema:

```json
{
  "user_id": "user_12345",
  "session_id": "session_67890",
  "recommendation_id": "rec_45678",
  "feedback_type": "property_rating",
  "feedback_data": {
    "property_id": "prop_98765",
    "user_rating": 4,
    "feedback_categories": {
      "accuracy": 5,
      "relevance": 4,
      "explanation_quality": 4
    },
    "comments": "Great match, but would prefer larger yard",
    "action_taken": "saved_property"
  }
}
```

AI & Agent Integration

Multi-Agent Architecture

Our system employs specialized AI agents that collaborate to provide comprehensive recommendations:

1. Budget Agent

- **Responsibility**: Financial analysis and affordability assessment
- **Capabilities**:
  - Market value analysis
  - Financing options evaluation
  - Price trend prediction
  - ROI calculations

2. Location Agent

- **Responsibility**: Geographic and commute optimization
- **Capabilities**:
  - Commute time calculation (real-time traffic)
  - Neighborhood analysis

    - School district evaluation
    - Crime and safety data integration

3. Features Agent

- **Responsibility**: Property amenities and lifestyle matching
- **Capabilities**:
    - Feature importance ranking
    - Lifestyle compatibility scoring
    - Maintenance cost estimation
    - Future upgrade potential

4. Coordinator Agent (Azure OpenAI GPT-4)

- **Responsibility**: Synthesize agent outputs and generate explanations
- **Capabilities**:
    - Natural language explanation generation
    - Conflict resolution between agent recommendations
    - User query interpretation
    - Contextual conversation management

Agent Communication Flow

```python
```

```
# Simplified agent coordination logic
class PropertyRecommendationOrchestrator:
    def __init__(self):
        self.budget_agent = BudgetAnalysisAgent()
        self.location_agent = LocationAnalysisAgent()
        self.features_agent = FeaturesAnalysisAgent()
        self.coordinator = GPT4CoordinatorAgent()

    async def get_recommendations(self, user_preferences):
        # Parallel agent execution
        budget_analysis = await self.budget_agent.analyze(user_preferences)
        location_analysis = await self.location_agent.analyze(user_preferences)
        features_analysis = await self.features_agent.analyze(user_preferences)

        # Coordinator synthesizes results
        recommendations = await self.coordinator.synthesize({
            'budget': budget_analysis,
            'location': location_analysis,
            'features': features_analysis,
            'user_context': user_preferences
        })

        return recommendations
```

Edge Case Handling

1. Timeout and Fallback Behavior

python

```python
# Timeout handling with circuit breaker pattern
from circuitbreaker import circuit

@circuit(failure_threshold=5, recovery_timeout=30)
async def get_ai_explanation(property_data, user_preferences):
    try:
        # Primary AI explanation generation
        return await openai_service.generate_explanation(
            property_data, user_preferences, timeout=10
        )
    except TimeoutError:
        # Fallback to rule-based explanation
        return generate_rule_based_explanation(property_data, user_preferences)
    except Exception:
        # Minimal safe explanation
        return "This property matches your basic requirements."
```

2. Rate Limiting and Abuse Prevention

python

```python
# Redis-based rate limiting
from fastapi import HTTPException
from redis import Redis

class RateLimiter:
    def __init__(self, redis_client: Redis):
        self.redis = redis_client

    async def check_rate_limit(self, user_id: str, endpoint: str):
        key = f"rate_limit:{user_id}:{endpoint}"
        current_requests = await self.redis.get(key)

        if current_requests and int(current_requests) > 100:  # 100 requests per hour
            raise HTTPException(
                status_code=429,
                detail="Rate limit exceeded. Try again later."
            )

        await self.redis.incr(key)
        await self.redis.expire(key, 3600)  # 1 hour window
```

3. Data Validation and Security

python

```python
from pydantic import BaseModel, validator
from typing import Optional, List

class UserPreferences(BaseModel):
    budget_min: int
    budget_max: int
    city: Optional[str] = None
    min_bedrooms: int
    max_commute_time_minutes: int

    @validator('budget_min', 'budget_max')
    def validate_budget(cls, v):
        if v < 0 or v > 50_000_000:  # Reasonable bounds
            raise ValueError('Budget must be between $0 and $50M')
        return v

    @validator('budget_max')
    def budget_max_greater_than_min(cls, v, values):
        if 'budget_min' in values and v <= values['budget_min']:
            raise ValueError('Maximum budget must be greater than minimum')
        return v

    @validator('city')
    def validate_city(cls, v):
        if v and len(v) > 50:
            raise ValueError('City name too long')
        return v
```

4. Monitoring and Alerting

python

```
# Application Insights integration
from applicationinsights import TelemetryClient
from datetime import datetime

class RecommendationMonitor:
    def __init__(self):
        self.telemetry = TelemetryClient(instrumentation_key="your-key")

    def track_recommendation_request(self, user_id, preferences, results):
        self.telemetry.track_event('RecommendationGenerated', {
            'user_id': user_id,
            'results_count': len(results),
            'processing_time': datetime.now().isoformat(),
            'city_filter': preferences.get('city', 'any')
        })

    def track_error(self, error_type, error_message, user_context):
        self.telemetry.track_exception(
            type=error_type,
            value=error_message,
            properties=user_context
        )
```

Security Best Practices

1. Authentication & Authorization

- Azure AD B2C for user authentication
- JWT tokens with short expiration (15 minutes)
- Refresh token rotation
- Role-based access control (RBAC)

2. Data Protection

- Encryption at rest (Azure Storage encryption)
- Encryption in transit (TLS 1.3)
- PII data masking in logs
- GDPR compliance for EU users

3. API Security

- Input validation and sanitization
- SQL injection prevention

- Rate limiting per user/IP
- CORS policy configuration
- API key management via Azure Key Vault

4. Infrastructure Security

- Network security groups (NSGs)
- Private endpoints for Azure services
- Web Application Firewall (WAF)
- DDoS protection
- Security baselines and compliance scanning

Scalability Considerations

1. Horizontal Scaling

- Auto-scaling App Services based on CPU/memory
- Container orchestration with Azure Kubernetes Service
- Database read replicas for query performance
- Redis cache for session management

2. Performance Optimization

- CDN for static assets (property images)
- Database indexing strategy
- Async processing for heavy operations
- Connection pooling for database connections

3. Cost Optimization

- Reserved instances for predictable workloads
- Spot instances for batch processing
- Storage tiering for historical data
- Function Apps for sporadic tasks

This architecture provides a robust, scalable foundation for the AI-powered property recommendation system while maintaining security, performance, and cost-effectiveness.