ThinkRealty Backend Developer Assessment

Ownership Transfer System Implementation

Time Allocation: 3 Days

Tech Stack: FastAPI + PostgreSQL + Docker + Redis

Background Context

You are building the ownership transfer system for ThinkRealty, a Dubai-based real estate platform. This system must handle complex ownership scenarios common in UAE real estate, including joint ownership, corporate ownership, inheritance transfers, and court-ordered transfers.

The system must maintain complete historical records, ensure data integrity, and handle real-time notifications for ownership changes.

Assessment Tasks

Task 1: Database Schema Design

Design and implement the complete database schema for the ownership transfer system.

Requirements:

1. **Core Tables to Design:**
    - `units` - Property units (pre-existing, minimal schema provided)
    - `owners` - Individual and corporate owners
    - `ownership_history` - Complete ownership timeline
    - `ownership_transfers` - Transfer transactions
    - `transfer_documents` - Legal documents
    - `audit_logs` - System audit trail
2. **Business Rules to Implement in Schema:**
    - Joint ownership: Multiple owners can own percentages of a single unit
    - Ownership percentages must always sum to 100% per unit
    - Corporate owners must have additional business fields
    - Individual owners require Emirates ID validation

- All ownership changes must be auditable
- No ownership record can be deleted (soft deletes only)

3. **Specific Schema Requirements:**

```
-- Starter schema (you need to complete this)
CREATE TABLE units (
    unit_id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    unique_key VARCHAR(255) UNIQUE NOT NULL,
    building_name VARCHAR(255) NOT NULL,
    unit_number VARCHAR(50) NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

4. **Complex Constraints to Implement:**

- Ensure ownership percentages don't exceed 100% per unit
- Validate Emirates ID format (784-YYYY-XXXXXXX-X)
- Prevent overlapping ownership periods for same owner-unit
- Ensure transfer dates are logical and sequential

Deliverables:

- Complete SQL schema with all tables, constraints, and indexes
- Database migration scripts
- Data validation triggers

Task 2: Core API Endpoints

Implement the following FastAPI endpoints with complete business logic:

2.1 Transfer Initiation Endpoint

```
POST /api/v1/ownership/transfers/initiate
```

Request Body:

```
{
    "unit_id": "uuid",
    "transfer_type": "purchase|inheritance|gift|court_order|corporate_res
    "current_owners": [
        {
            "owner_id": "uuid",
```

```
            "ownership_percentage": 100.0,
            "transfer_percentage": 50.0
        }
    ],
    "new_owners": [
        {
            "owner_data": {
                "full_name": "Ahmed Al Mansouri",
                "emirates_id": "784-2024-1234567-8",
                "phone": "+971501234567",
                "owner_type": "individual|corporate"
            },
            "ownership_percentage": 50.0
        }
    ],
    "transfer_date": "2024-01-15",
    "purchase_price": 1500000.00,
    "legal_reason": "Purchase agreement",
    "documents": ["document_id_1", "document_id_2"]
}
```

Business Logic Requirements:

- Validate all ownership percentages sum correctly
- Check for existing ownership conflicts
- Validate Emirates ID format and uniqueness
- Ensure transfer doesn't create ownership gaps
- Handle partial transfers (owner keeps remaining percentage)
- Create audit trail entries
- Cache ownership data in Redis for performance

2.2 Complex Portfolio Query Endpoint

```
GET /api/v1/owners/{owner_id}/portfolio
```

Query Parameters:

- `include_history` : boolean (default: false)
- `date_range` : "YYYY-MM-DD to YYYY-MM-DD"
- `status_filter` : "current|historical|all"

Response Requirements:

- Complete ownership timeline for the owner
- Current portfolio value estimation
- Historical transaction summary
- Joint ownership details
- Performance metrics (ROI, holding periods)

2.3 Ownership Validation Endpoint

```
POST /api/v1/ownership/validate-transfer
```

Purpose: Validate a proposed transfer before execution
Requirements:

- Check all business rules without saving data
- Return detailed validation results
- Identify potential conflicts or issues
- Suggest corrections for invalid transfers

Task 3: Complex Business Logic Implementation

3.1 Joint Ownership Handler

Implement a service class `JointOwnershipManager` that handles:

```python
class JointOwnershipManager:
    async def split_ownership(
        self,
        unit_id: UUID,
        current_owner_id: UUID,
        split_percentages: List[Dict[str, Any]]
    ) -> TransferResult:
        """
        Split ownership among multiple new owners
        Example: 100% owner splits to 60% + 40% between two people
        """
        pass

    async def consolidate_ownership(
        self,
        unit_id: UUID,
        owners_to_consolidate: List[UUID],
        new_owner_id: UUID
    ) -> TransferResult:
```

```
        """
        Consolidate multiple ownerships into single ownership
        Example: 30% + 70% owners sell to single new owner
        """
        pass

    async def redistribute_ownership(
        self,
        unit_id: UUID,
        new_distribution: Dict[UUID, float]
    ) -> TransferResult:
        """
        Redistribute percentages among existing owners
        Example: 3 owners (40%, 30%, 30%) -> (50%, 25%, 25%)
        """
        pass
```

3.2 Inheritance Transfer Logic

Implement inheritance-specific business rules:

- Handle deceased owner transfers
- Validate heir relationships
- Implement Islamic inheritance law calculations (if applicable)
- Handle probate court requirements
- Manage temporary ownership during probate

3.3 Corporate Ownership Handler

Implement corporate-specific ownership rules:

- Company ownership percentage limits
- Shareholder change notifications
- Corporate restructuring transfers
- Multiple signatory requirements
- Board resolution requirements

Task 4: Advanced Query Implementation

Write complex PostgreSQL queries for these scenarios:

4.1 Portfolio Analytics Query

```
-- Find all owners who have increased their portfolio value by >50%
-- in the last 2 years through purchases/sales, including:
-- - Original investment amount
-- - Current portfolio value
-- - Percentage gain
-- - Number of transactions
-- - Average holding period
```

4.2 Ownership Conflict Detection Query

```
-- Identify potential ownership conflicts:
-- - Units with ownership percentages not summing to 100%
-- - Overlapping ownership periods for same owner-unit
-- - Ownership transfers without proper documentation
-- - Owners with suspicious transaction patterns
```

4.3 Market Analysis Query

```
-- Generate ownership transfer trends report:
-- - Monthly transfer volumes by property type
-- - Average transfer values by area
-- - Most active owners (buyers/sellers)
-- - Ownership concentration analysis
```

Task 5: Error Handling & Edge Cases

Implement comprehensive error handling for these scenarios:

1. **Concurrent Transfer Attempts:** Two transfers initiated simultaneously for same unit
2. **Invalid Percentage Distributions:** Percentages don't sum to 100%
3. **Missing Legal Documents:** Required documents not provided
4. **Database Constraint Violations:** Foreign key or unique constraint errors
5. **External Service Failures:** Document verification service unavailable
6. **Rollback Scenarios:** Partial transfer completion requiring rollback

Task 6: Caching Strategy Implementation

Implement Redis caching for:

1. **Owner Portfolio Cache:** Cache frequently accessed portfolio data
2. **Ownership Validation Cache:** Cache validation results for duplicate checks

3. **Transfer Status Cache:** Real-time transfer status updates
4. **Audit Log Cache:** Recent audit entries for quick access

Requirements:

- Implement cache invalidation strategies
- Handle cache warming for critical data
- Implement fallback mechanisms when cache is unavailable

Technical Specifications

Docker Setup Required

```
# Provide basic Dockerfile structure
FROM python:3.11-slim
# Complete the rest...
```

Environment Configuration

```
DATABASE_URL=postgresql://user:pass@postgres:5432/thinkrealty
REDIS_URL=redis://redis:6379/0
```

Testing Requirements

Write unit tests for:

- Ownership percentage validation
- Transfer business logic
- Edge case handling
- Database constraint enforcement

Evaluation Criteria

Code Quality (25%)

- Clean, readable, well-documented code
- Proper separation of concerns
- Consistent naming conventions
- Appropriate error handling

Database Design (25%)

- Proper normalization and relationships
- Efficient indexing strategy
- Constraint implementation
- Performance considerations

Business Logic (25%)

- Correct implementation of ownership rules
- Proper handling of edge cases
- Validation logic completeness
- Audit trail implementation

API Design (15%)

- RESTful API principles
- Proper HTTP status codes
- Input validation and sanitization
- Response structure consistency

Performance & Scalability (10%)

- Efficient query design
- Caching implementation
- Database optimization
- Concurrent request handling

Submission Requirements

1. **Complete Source Code**
    - All Python files with FastAPI implementation
    - Database migration scripts
    - Docker configuration files
    - Requirements.txt with dependencies
2. **Database Schema**
    - Complete SQL schema file
    - Migration scripts (up/down)
    - Sample data inserts
3. **Documentation**
    - API documentation (can use FastAPI auto-docs)

- Database schema documentation
- Business rules explanation
- Setup and running instructions

4. **Test Cases**
    - Unit tests for core business logic
    - Integration tests for API endpoints
    - Edge case test scenarios

Prohibited Resources

- No AI assistants (ChatGPT, Copilot, etc.)
- No direct code copying from online sources
- You may use official documentation only
- Standard libraries and framework docs are allowed

Sample Test Scenarios

Test your implementation with these scenarios:

1. **Ahmed owns 100% of Unit A, sells 30% to Sarah and 20% to Omar**
2. **Corporate owner XYZ Ltd (60%) and individual owner Fatima (40%) sell entire unit to new buyer**
3. **Owner dies, inheritance split among 3 heirs with Islamic law percentages**
4. **Court orders forced sale of jointly owned property**
5. **Company restructuring requiring ownership transfer between subsidiaries**

Good luck!