
Department of Information and Computer Science

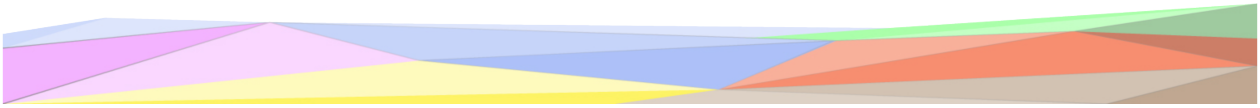
National University of Mongolia

# KGE-MN 2025 - Knowledge Graph utilizing open data from the State Registration of Legal Entities of Mongolia

| Document Data: | Reference Persons: |
| --- | --- |
| December 15, 2025 | Chinzorigt.G, Enkhbayasgalan.E |

Ulaanbaatar, Mongolia





# Contents

# Revision History:

| Revision | Date | Author | Description of Changes |
|---|---|---|---|
| 0.1 | December 2, 2025 | Chinzorigt.G | Document created |
| 0.2 | December 7, 2025 | Chinzorigt.G | Introduction & DOI section created |
| 0.3 | December 10, 2025 | Chinzorigt.G | PFsheet (XLSX) & ER Model (PNG) created |
| 0.4 | December 14, 2025 | Chinzorigt.G | Purpose Formalization section created |
| 0.5 | December 14, 2025 | Enkhbayasgalan.E | Coding web scraping and JSON data extraction |
| 0.6 | December 14, 2025 | Chinzorigt.G | Information gathering section created |

# 1 Introduction

Reusability is one of the main principles in the Knowledge Graph Engineering (KGE) process defined by iTelos. The KGE project documentation plays an important role in enhancing the reusability of the resources handled and produced during the process. A clear description of the resources as well as of the process (and sub-processes) developed, provides a clear understanding of the project, thus serving such information to external readers for the future exploitation of the project's outcomes.

## 1.1 Project Overview

This project focuses on the construction of a Knowledge Graph utilizing open data from the State Registration of Legal Entities of Mongolia. The primary objective is to visualize, in a graph structure, the complex relationships among various stakeholders within the Mongolian corporate ecosystem. Specifically, the project addresses the following relationship categories:

- **Shareholders and Members**: Individuals and entities holding ownership stakes in legal entities, including their classification and country of origin.
- **Officials and Controlling Entities**: Persons authorized to represent legal entities without a power of attorney, including their official positions and appointment dates.
- **Ultimate Beneficial Owners**: Natural persons who ultimately own or control legal entities, enabling transparency in corporate ownership structures.

Furthermore, for companies listed on the Mongolian Stock Exchange, the project aims to link and visualize relevant open datasets, creating an integrated view of corporate information that spans both registration data and capital market participation.

## 1.2 Motivation and Significance

The transparency of corporate ownership structures is essential for various stakeholders, including regulatory authorities, financial institutions, investors, and the general public. In Mongolia, as in many jurisdictions, complex corporate structures can obscure the true ownership and control of legal entities. This Knowledge Graph addresses this challenge by:

- Enabling the visualization of multi-layered ownership networks
- Facilitating the identification of individuals with significant influence across multiple entities

- Supporting regulatory compliance and anti-money laundering efforts
- Providing investors and business partners with comprehensive due diligence capabilities

### 1.3 Document Structure

The current document aims to provide a detailed report of the project developed following the iTelos methodology. The report is structured as follows:

- **Section 2**: Definition of the project's purpose and its domain of interest, establishing the scope and objectives that guide all subsequent development activities.
- **Section 3**: High-level description of the project development, based on the Produce role's objectives, providing an overview of the production strategy and key milestones.
- **Sections 4, 5, 6, 7, and 8**: The description of the iTelos process phases and their activities, divided by knowledge and data layer activities. These sections detail the systematic approach taken to formalize the purpose, design the knowledge architecture, and implement the data integration processes.
- **Section 9**: The description of the evaluation criteria and metrics applied to the project's final outcome, ensuring the quality and fitness-for-purpose of the resulting Knowledge Graph.
- **Section 10**: The description of the metadata produced for all (and all kinds of) resources handled and generated by the iTelos process while executing the project, supporting long-term maintainability and reusability.
- **Section 11**: Conclusions and open issues summary, reflecting on the project outcomes and identifying opportunities for future development and enhancement.

# 2 Domain of Interest (DoI)

This section defines the boundaries of the Knowledge Graph Engineering project in terms of spatial and temporal dimensions. The Domain of Interest establishes the scope within which the project purpose—visualizing relationships among shareholders, officials, controlling entities, and ultimate beneficial owners of Mongolian legal entities—will be realized.

## 2.1 Spatial Boundaries

### 2.1.1 Primary Geographic Scope

The Domain of Interest is geographically bounded to **Mongolia**, specifically encompassing:

- **National Coverage**: All legal entities registered with the State Registration of Legal Entities of Mongolia, regardless of their physical location within the country's 21 aimags (provinces) and the capital city of Ulaanbaatar.
- **Administrative Divisions**: The registered addresses of legal entities span all administrative levels, including:
  - Ulaanbaatar (capital city) and its districts (düüregs)
  - Provincial capitals (aimag centers)
  - District subdivisions (khoroos and bags)

### 2.1.2 International Dimension

While the primary focus is Mongolia, the domain necessarily extends to include international elements due to the nature of corporate ownership:

- **Foreign Shareholders**: Legal entities may have shareholders from foreign countries (e.g., Singapore, Hungary, as seen in the example data with "Хайнекен Азиа Пасифик Пте Лтд" from Singapore and "Steppe Beverage KFT" from Hungary). The Knowledge Graph will capture the country of origin for these foreign stakeholders.
- **Cross-Border Ownership Chains**: The graph will represent ownership relationships that cross national boundaries, though detailed information about foreign parent companies is limited to what is recorded in the Mongolian registry.
- **Boundary Limitation**: The project does not extend to foreign corporate registries. Information about foreign shareholders is limited to their name, country of origin, and relationship to Mongolian entities as recorded in the State Registration system.

### 2.1.3 Spatial Boundary Justification

The geographic boundaries were defined based on the following considerations:

1. **Data Availability**: The open data from the State Registration of Legal Entities covers all legally registered entities within Mongolia's jurisdiction, providing comprehensive national coverage.

2. **Legal Framework**: The Mongolian Company Law and relevant regulations govern entities within these boundaries, ensuring data consistency and regulatory compliance.

3. **User Needs**: The identified personas (investigators, business analysts, portfolio managers, compliance officers) primarily operate within the Mongolian legal and business environment, making national scope most relevant to their needs.

## 2.2 Temporal Boundaries

### 2.2.1 Historical Scope

The temporal dimension of the Domain of Interest encompasses:

- **Start Date**: The Knowledge Graph will include data from **January 1, 2000** onwards. This date was selected because:
  - It captures the modern era of Mongolia's market economy development
  - Most currently active legal entities were registered after this date
  - Data quality and completeness improve significantly from this period
- **End Date**: The temporal scope extends to the **present day**, with the expectation of ongoing updates as new registrations and changes occur in the source registry.
- **Historical Records**: For entities registered before 2000 that remain active, their historical information (as available in the registry) will be included, though with the understanding that older records may be less complete.

### 2.2.2 Temporal Granularity

The Knowledge Graph captures temporal information at the following levels of granularity:

- **Registration Dates**: Precise dates (YYYY.MM.DD format) for:
  - Initial entity registration
  - Shareholder/member registration
  - Appointment of authorized representatives
  - Ultimate beneficial owner registration
  - Business activity registration
  - Restructuring events

- **Validity Periods**: For certain business activities (particularly licensed activities such as alcohol production), the data includes validity periods with start and end dates (e.g., ”2019.04.14 - 2022.04.14” for alcohol production licenses).
- **Change Tracking**: The system captures the dates when changes occurred, enabling temporal analysis of corporate evolution.

#### 2.2.3 Temporal Boundary Justification

The temporal boundaries were established based on:

1. **Data Completeness**: Records from 2000 onwards demonstrate higher data quality and completeness compared to earlier periods.
2. **Relevance to Current Analysis**: The 20+ year historical window provides sufficient depth for:
   - Tracking corporate evolution and restructuring
   - Identifying long-term patterns in ownership and control
   - Supporting due diligence investigations requiring historical context
3. **Regulatory Evolution**: Mongolia’s modern corporate governance framework, including beneficial ownership disclosure requirements, has developed primarily within this timeframe.

### 2.3 Domain Boundary Summary

| Dimension | Boundary Definition |
|---|---|
| **Geographic Scope** | Mongolia (all 21 aimags and Ulaanbaatar) |
| **International Elements** | Foreign shareholder countries (as recorded in Mongolian registry) |
| **Institutional Scope** | State Registration of Legal Entities, Mongolian Stock Exchange |
| **Temporal Start** | January 1, 2000 |
| **Temporal End** | Present (with ongoing updates) |
| **Temporal Granularity** | Daily (date-level precision for all recorded events) |

Table 1: Summary of Domain of Interest Boundaries

### 2.4 Out of Scope

To provide clarity on the Domain of Interest boundaries, the following elements are explicitly **excluded** from the project scope:

- **Foreign Registry Data**: Detailed corporate information from foreign jurisdictions (beyond what is recorded in the Mongolian registry)

- **Informal Enterprises**: Unregistered businesses or sole proprietorships not captured in the State Registration system
- **Historical Records Pre-1990**: Data from the socialist period before Mongolia's transition to a market economy
- **Non-Corporate Entities**: Government agencies, international organizations, and diplomatic missions (unless they appear as shareholders in registered companies)
- **Real-Time Transaction Data**: Stock trading data, financial transactions, or other real-time market information beyond static company registration data

# 3 Project Development

This section describes, at top level, how the project's objectives (or "The Purpose") will be satisfied. More in details the current section aims at describing how the dta production process is performed.

## 3.1 Data Production

The description of which (quality) data needs to be created to satisfy the project purpose. This sub-section highlights the role of the data producer. The sub-section aims at describing how the data producer creates the data required to satisfy the project's purpose.

# 4 Initial Resources

This section describes the already available resources considered for the project. More in detail the resources here described, are quality resources (compliant with the quality and reusability guidelines defined by iTleos. 6*, or at least 5*) which don't need to be processed or created by a data producer. The resources described in this section are those that can be already composed by the data consumer to satisfy the project's purpose.

In this section are described both the resourced selected, and the sources from which such resources have been retrieved.

This section describes the two kind of resources considered by a projects, by filling the two sub-sections here below.

- **Knowledge resources**: iTelos compliant reference schemas and ontologies initially collected to satisfy the purpose along the KGE process. The knowledge resources initial metadata have to be reported here.
- **Data sources**: iTelos compliant datasets initially collected to satisfy the purpose along the KGE process. The data resources initial metadata have to be reported here.

# 5 Purpose Formalization

This section documents the activities and results achieved during the first phase of the iTelos methodology for constructing a Knowledge Graph based on the State Registration of Legal Entities in Mongolia. The project aims to visualize relationships among shareholders, members, officials, controlling entities, and ultimate beneficial owners, with additional linkages to Mongolian Stock Exchange data for listed companies.

## 5.1 Scenarios Definition

The following usage scenarios describe the multiple aspects considered by the project purpose:

### 5.1.1 Scenario 1: Corporate Ownership Investigation

A financial investigator needs to trace the ownership structure of a company suspected of involvement in financial irregularities. The investigator must identify all shareholders, their respective ownership percentages, and any connections to other legal entities. The system should reveal complex ownership chains, including nested corporate structures where companies own shares in other companies, ultimately leading to the identification of ultimate beneficial owners.

### 5.1.2 Scenario 2: Due Diligence for Business Partnerships

A business development manager at a Mongolian corporation is evaluating potential partners for a joint venture. Before entering negotiations, they need to understand the governance structure of target companies, including who has authority to represent the company without power of attorney, the company's business activities, and any organizational restructuring history that might indicate instability or strategic pivots.

### 5.1.3 Scenario 3: Stock Market Investment Analysis

An investment analyst researching publicly traded companies on the Mongolian Stock Exchange requires comprehensive information about company leadership, ownership con-

centration, and cross-holdings between listed entities. The analyst needs to identify potential conflicts of interest where the same individuals serve as officials across multiple companies or where significant ownership overlaps exist.

#### 5.1.4 Scenario 4: Regulatory Compliance Monitoring

A compliance officer at a regulatory authority monitors legal entities for adherence to ownership disclosure requirements. They need to identify companies where ultimate beneficial owner information is incomplete, track changes in controlling persons over time, and detect patterns that might indicate attempts to obscure true ownership.

#### 5.1.5 Scenario 5: Anti-Money Laundering (AML) Analysis

An AML specialist investigates networks of companies that may be used for layering illicit funds. The specialist needs to visualize connections between entities through shared shareholders, officials, and beneficial owners, identifying clusters of related companies and individuals who appear across multiple entities in patterns suggesting coordinated control.

### 5.2 Personas

#### 5.2.1 Persona 1: Investigator Batbold

- **Role:** Financial Crimes Investigator at the Financial Regulatory Commission
- **Age:** 42 years old
- **Background:** 15 years of experience in financial investigation, former police detective
- **Technical Skills:** Moderate; comfortable with databases but prefers visual interfaces
- **Goals:** Quickly identify ownership networks, trace beneficial owners, and document evidence chains for legal proceedings
- **Pain Points:** Currently relies on manual searches through multiple registries; difficulty connecting entities across different data sources
- **Primary Scenario:** Scenario 1, Scenario 5

#### 5.2.2 Persona 2: Business Analyst Oyungerel

- **Role:** Senior Business Development Manager at a mining corporation
- **Age:** 35 years old
- **Background:** MBA graduate, 10 years in corporate strategy
- **Technical Skills:** High; proficient in data analysis tools and visualization platforms
- **Goals:** Conduct thorough due diligence on potential partners, understand corporate governance structures, assess business stability
- **Pain Points:** Time-consuming process to gather information from multiple sources; difficulty assessing company history and restructuring events
- **Primary Scenario:** Scenario 2

#### 5.2.3 Persona 3: Portfolio Manager Enkhjargal

- **Role:** Portfolio Manager at an investment fund
- **Age:** 38 years old
- **Background:** CFA charterholder, specializes in Mongolian equities
- **Technical Skills:** Very high; uses quantitative analysis tools daily
- **Goals:** Identify investment opportunities, assess governance risks, understand ownership concentration in listed companies
- **Pain Points:** Limited integration between stock exchange data and corporate registry information; manual effort required to build comprehensive company profiles
- **Primary Scenario:** Scenario 3

#### 5.2.4 Persona 4: Compliance Officer Munkhbat

- **Role:** Senior Compliance Officer at the General Authority for State Registration
- **Age:** 45 years old
- **Background:** Legal background, 20 years in public administration
- **Technical Skills:** Moderate; familiar with government databases
- **Goals:** Monitor compliance with disclosure requirements, identify incomplete registrations, generate compliance reports

- **Pain Points:** Difficulty tracking changes over time; no automated alerting for suspicious patterns
- **Primary Scenario:** Scenario 4

## 5.3 Competency Questions (CQs)

The following competency questions were developed based on the personas and scenarios defined above:

### 5.3.1 Entity Identification and Basic Information

CQ1: What is the registration number of a legal entity given its name?

CQ2: What is the registration date of a specific legal entity?

CQ3: What is the legal form (Хэлбэр) of a given company?

CQ4: What is the type (Төрөл) classification of a legal entity?

CQ5: What is the registered address of a legal entity?

### 5.3.2 Ownership and Shareholder Information

CQ6: Who are all the shareholders and members of a specific legal entity?

CQ7: What is the classification (Ангилал) of each shareholder in a company?

CQ8: Which country does each shareholder belong to?

CQ9: What is the gender distribution of shareholders in a given company?

CQ10: When was each shareholder registered as a member of the company?

CQ11: Which companies share common shareholders?

CQ12: What legal entities does a specific individual hold shares in?

### 5.3.3 Management and Representation Authority

CQ13: Who are the officials authorized to represent a company without power of attorney?

CQ14: What position does each authorized representative hold?

CQ15: Which companies does a specific individual have authority to represent?

CQ16: Are there individuals who serve as authorized representatives in multiple companies?

CQ17: What is the registration date of each authorized representative's appointment?

#### 5.3.4 Ultimate Beneficial Ownership

CQ18: Who are the ultimate beneficial owners of a specific legal entity?

CQ19: What is the classification of each ultimate beneficial owner?

CQ20: Which companies share the same ultimate beneficial owner?

CQ21: For a given individual, in which companies are they listed as an ultimate beneficial owner?

CQ22: Which companies lack complete ultimate beneficial owner information?

#### 5.3.5 Business Activities

CQ23: What are the registered business activities of a legal entity?

CQ24: What is the status (active/inactive) of each business activity?

CQ25: Which companies operate in the same business sector?

CQ26: When was a specific business activity registered for a company?

#### 5.3.6 Corporate Restructuring and History

CQ27: Has a legal entity undergone any organizational restructuring?

CQ28: What was the previous name of a restructured company?

CQ29: What type of restructuring occurred (merger, division, transformation)?

CQ30: What is the chronological history of changes for a given company?

#### 5.3.7 Network Analysis and Cross-Entity Queries

CQ31: What is the network of companies connected through shared ownership?

CQ32: Which individuals appear in multiple roles (shareholder, official, beneficial owner) across different companies?

CQ33: What is the degree of separation between two legal entities through ownership or management relationships?

CQ34: Which clusters of companies exhibit patterns of coordinated control?

## 5.4 Concepts Identification

Based on the scenarios, personas, and competency questions, the following concepts have been identified and classified according to their popularity and relevance to the project purpose.

### 5.4.1 High Popularity Concepts (Core Entities)

| Concept (English) | Concept (Mongolian) | Description |
|---|---|---|
| Legal Entity | Хуулийн этгээд | The primary entity representing registered companies and organizations |
| Person | Хүн | Individual persons who can be shareholders, officials, or beneficial owners |
| Shareholder/Member | Хувьцаа эзэмшигч/Гишүүн | Persons or entities holding ownership stakes |
| Authorized Representative | Итгэмжлэлгүй төлөөлөгч | Officials with authority to represent without power of attorney |
| Ultimate Beneficial Owner | Эцсийн өмчлөгч | The final natural person who ultimately owns or controls the entity |

### 5.4.2 Medium Popularity Concepts (Supporting Entities)

| Concept (English) | Concept (Mongolian) | Description |
|---|---|---|
| Business Activity | Үйл ажиллагааны чиглэл | Types of business operations registered for an entity |
| Position/Title | Албан тушаал | Official positions held by authorized representatives |
| Legal Form | Хэлбэр | The legal structure type of the entity (LLC, JSC, etc.) |
| Entity Type | Төрөл | Classification type of the legal entity |

| Concept (English) | Concept (Mongolian) | Description |
|---|---|---|
| Country | Улс | Country of origin for foreign shareholders |

#### 5.4.3 Low Popularity Concepts (Contextual Information)

| Concept (English) | Concept (Mongolian) | Description |
|---|---|---|
| Restructuring Event | Өөрчлөлт | Corporate reorganization events |
| Address | Хаяг | Physical location of the legal entity |
| Classification | Ангилал | Category classification for shareholders and beneficial owners |
| Activity Status | Төлөв | Current status of business activities |

#### 5.4.4 Property Identification

| Property (English) | Property (Mongolian) | Data Type | Applies To |
|---|---|---|---|
| Registration Number | Регистрийн дугаар | Number | Legal Entity |
| Name | Оноосон нэр | String | Legal Entity |
| Registration Date | Бүртгэсэн огноо | Date | Multiple entities |
| First Name | Нэр | String | Person |
| Patronymic | Эцэг/эх/-ийн нэр | String | Person |
| Gender | Хүйс | String | Person |
| Country Name | Улсын нэр | String | Person |
| Position Title | Албан тушаал | String | Authorized Representative |
| Activity Direction | Үйл ажиллагааны чиглэл | String | Business Activity |
| Status | Төлөв | String | Business Activity |

| Property (English) | Property (Mongolian) | Data Type | Applies To |
|---|---|---|---|
| Restructuring Type | Өөрчлөгдөн зохион байгуулсан хэлбэр | String | Restructuring Event |
| Previous Name | Өөрчлөлтийн өмнөх оноосон нэр | String | Restructuring Event |
| Change Notes | Өөрчлөлтийн тэмдэглэл | String | Restructuring Event |

## 5.5 ER Model Definition

Based on the concepts and properties identified above, the following Entity-Relationship model has been designed to represent the purpose of the Knowledge Graph.

### 5.5.1 Entity Descriptions

**LegalEntity** - The central entity representing registered legal entities in Mongolia.

- Primary Key: registrationNumber
- Attributes: name, registrationDate, legalForm, entityType, address

**Person** - Represents natural persons who participate in legal entities.

- Attributes: firstName, patronymic, gender, countryName

**BusinessActivity** - Represents registered business activity directions.

- Attributes: activityDirection, status, registrationDate

**Position** - Represents official positions/titles.

- Attributes: positionTitle

**RestructuringEvent** - Represents corporate reorganization events.

- Attributes: restructuringType, registrationDate, previousName, changeNotes

### 5.5.2 Relationship Descriptions

| Relationship | From Entity | To Entity | Cardinality | Attributes |
|---|---|---|---|---|
| hasShareholder | LegalEntity | Person | 1:N | classification, registrationDate |
| hasAuthorizedRep | LegalEntity | Person | 1:N | position, registrationDate |
| hasBeneficialOwner | LegalEntity | Person | 1:N | classification, registrationDate |
| hasActivity | LegalEntity | BusinessActivity | 1:N | - |
| hasRestructuring | LegalEntity | RestructuringEvent | 1:N | - |
| holdsPosition | Person | Position | N:M | - |

#### 5.5.3 ER Diagram

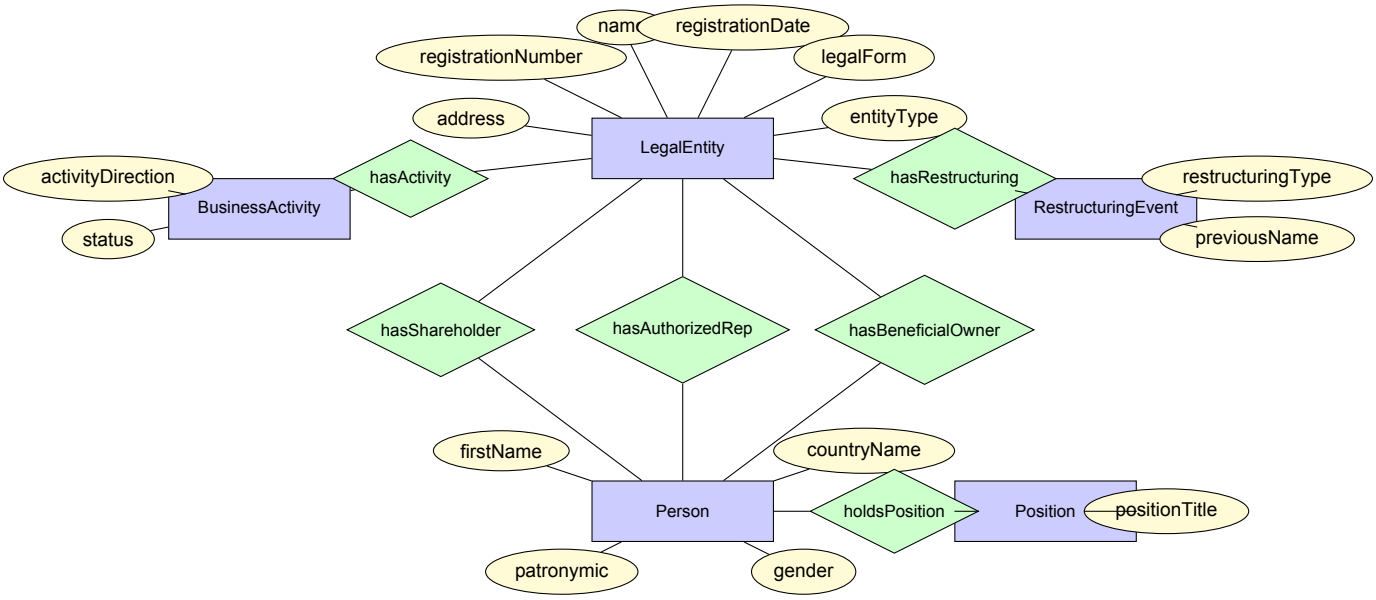



Figure 1: Entity-Relationship Diagram for Mongolian Legal Entities Knowledge Graph

### 5.6 Design Decisions and Rationale

#### 5.6.1 Strengths of the Proposed Model

1. **Comprehensive Coverage**: The model captures all key aspects of the source data, including ownership, management, beneficial ownership, business activities, and corporate history.

2. **Network Analysis Support**: The design enables complex graph queries to identify relationships between entities and persons across multiple companies, supporting the investigative and analytical use cases identified in the scenarios.
3. **Temporal Tracking**: Registration dates are captured for all major relationships, enabling historical analysis and change tracking.
4. **Flexibility for Extension**: The model can be extended to incorporate Mongolian Stock Exchange data through additional entities and relationships without major restructuring.

#### 5.6.2 Limitations and Trade-offs

1. **Person Identification**: The source data does not include unique identifiers for persons, making it challenging to definitively link the same individual across different roles and companies. Name-based matching may be required.
2. **Ownership Percentages**: The current data format does not explicitly include shareholding percentages, which limits quantitative ownership analysis.
3. **Historical Completeness**: Changes in shareholders, officials, and beneficial owners over time are not fully captured in the source data structure, limiting historical analysis capabilities.

#### 5.6.3 Alternative Approaches Considered

1. **Single Person Entity vs. Role-Specific Entities**: We considered creating separate entities for Shareholders, Officials, and Beneficial Owners. However, the unified Person entity was chosen to better support network analysis and identify individuals appearing in multiple roles.
2. **Address as Entity vs. Attribute**: Address could be modeled as a separate entity to enable geographic analysis. This was deferred to keep the initial model simpler, but remains a candidate for future extension.

# 6 Information Gathering

This section reports the execution of the activities involved in the Information Gathering iTelos phase. This section describes both the resources selected and the sources from which such resources have been retrieved. The project focuses on gathering data about Mongolian legal entities from government open data portals and supplementary information from the Mongolian Stock Exchange for publicly listed companies.

## 6.1 Knowledge Layer

### 6.1.1 Sources Description

The knowledge layer resources provide the conceptual foundation and domain understanding necessary for modeling Mongolian legal entity structures.

### 6.1.2 Informal Resources Collection

Knowledge resources were collected through manual review of documentation, legal frameworks, and domain expert consultations. Key terminological resources include:

- Legal entity classification schemas from the State Registration Authority
- Ownership structure definitions from Mongolian Company Law
- Financial reporting terminology from the Mongolian Stock Exchange
- Business activity classification codes (aligned with ISIC standards)

### 6.1.3 Knowledge Resources Classification

## 6.2 Data Layer

### 6.2.1 Sources Description

Two primary data sources were identified for this project:

### 6.2.2 Source 1: Mongolia Open Data Portal

**Description:** The Mongolia Open Data Portal (`https://opendata.gov.mn`) is the official government platform for publishing open datasets. The legal entities dataset (ID: 5372) contains comprehensive registration information including:

- Basic entity information (registration number, name, date, form, type, address)
- Shareholder and member information
- Authorized representatives (officials without power of attorney)
- Ultimate beneficial owners
- Business activity registrations
- Organizational restructuring history

**Access Method:** Direct JSON file download via API endpoint.
**Data Quality Assessment:**

Table 7: Knowledge Layer Sources

| Source Name | URL | Type | Description |
|---|---|---|---|
| Mongolia Open Data Portal | https://opendata.gov.mn | Government Portal | Official open data platform providing metadata and data dictionaries |
| General Authority for State Registration | https://burtgel.gov.mn | Government Website | Legal entity registration authority with terminology definitions |
| Mongolian Stock Exchange | https://mse.mn/mn | Financial Institution | Stock exchange with listed company information and financial terminology |
| Company Law of Mongolia | Legal Document | Legislation | Defines legal entity types, ownership structures, and governance requirements |

Table 8: Knowledge Resources Classification

| Resource | Classification | Justificeation |
|---|---|---|
| Legal Entity Types | Core | Fundamental classification for all registered entities |
| Person/Organization | Common | Standard distinction used across multiple domains |
| Ownership Relations | Core | Central to the project purpose of visualizing ownership |
| Business Activity Codes | Contextual | Supporting information for entity characterization |
| Restructuring Types | Contextual | Historical information for tracking changes |
| Stock Exchange Listings | Contextual | Additional data for publicly traded companies |

Table 9: Data Layer Sources

| Source | URL | Format | Access | Update Freq. |
|---|---|---|---|---|
| Mongolia Open Data - Legal Entities | https://opendata.gov.mn/dataview/5372 | JSON | Direct Download | Periodic |
| Mongolian Stock Exchange | https://mse.mn/mn | HTML | Web Scraping | Real-time |

- **Strengths:** Official government source, structured JSON format, comprehensive coverage, regular updates
- **Weaknesses:** Some fields may contain null values, Mongolian language requires encoding handling, no unique person identifiers

#### 6.2.3 Source 2: Mongolian Stock Exchange

**Description:** The Mongolian Stock Exchange (MSE) website provides information about publicly listed companies, including trading data, company profiles, financial reports, and corporate announcements.

**Access Method:** Web scraping required as no official API is available.

**Data Quality Assessment:**

- **Strengths:** Real-time market data, detailed company profiles for listed entities, financial performance metrics
- **Weaknesses:** Requires web scraping, HTML structure may change, limited to publicly traded companies only

#### 6.2.4 Resources Collection and Scraping

**6.2.4.1 JSON Data Extraction from Open Data Portal** The following Python code extracts legal entity data from the Mongolia Open Data Portal:

```
import requests
import json
import os
from datetime import datetime
from typing import Optional, List, Dict, Any

class MongoliaOpenDataExtractor:
    """
    Extractor for Mongolia Open Data Portal - Legal Entities Dataset
    Source: https://opendata.gov.mn/dataview/5372
    """

    def __init__(self, output_dir: str = "data"):
        self.base_url = "https://opendata.gov.mn"
        self.dataset_id = "5372"
        self.output_dir = output_dir
        self.session = requests.Session()
        self.session.headers.update({
            'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit
    /537.36',
            'Accept': 'application/json',
```

```
            'Accept-Language': 'mn-MN,mn;q=0.9,en;q=0.8'
        })

        # Create output directory if not exists
        if not os.path.exists(output_dir):
            os.makedirs(output_dir)

    def get_dataset_metadata(self) -> Optional[Dict[str, Any]]:
        """Retrieve dataset metadata including download links"""
        try:
            metadata_url = f"{self.base_url}/api/3/action/package_show?id={self.dataset_id}"
            response = self.session.get(metadata_url, timeout=30)
            response.raise_for_status()
            return response.json()
        except requests.RequestException as e:
            print(f"Error fetching metadata: {e}")
            return None

    def download_json_data(self, url: str = None) -> Optional[List[Dict]]:
        """
        Download JSON data from the open data portal
        Direct download from dataview endpoint
        """
        try:
            # Direct download URL for dataset 5372
            if url is None:
                url = f"{self.base_url}/dataview/{self.dataset_id}"

            print(f"Downloading data from: {url}")
            response = self.session.get(url, timeout=120)
            response.raise_for_status()

            # Handle different response formats
            content_type = response.headers.get('Content-Type', '')

            if 'application/json' in content_type:
                data = response.json()
            else:
                # Try to parse as JSON anyway
                data = json.loads(response.text)

            print(f"Successfully downloaded {len(data) if isinstance(data, list) else 1} records")
            return data

        except requests.RequestException as e:
```

```
            print(f"Error downloading data: {e}")
            return None
        except json.JSONDecodeError as e:
            print(f"Error parsing JSON: {e}")
            return None

    def download_with_pagination(self, limit: int = 1000) -> List[Dict]:
        """
        Download data with pagination support for large datasets
        """
        all_data = []
        offset = 0

        while True:
            try:
                # API endpoint with pagination
                url = f"{self.base_url}/api/3/action/datastore_search"
                params = {
                    'resource_id': self.dataset_id,
                    'limit': limit,
                    'offset': offset
                }

                response = self.session.get(url, params=params, timeout=60)
                response.raise_for_status()
                result = response.json()

                if result.get('success'):
                    records = result.get('result', {}).get('records', [])
                    if not records:
                        break

                    all_data.extend(records)
                    print(f"Downloaded {len(all_data)} records...")
                    offset += limit
                else:
                    break

            except requests.RequestException as e:
                print(f"Error during pagination: {e}")
                break

        return all_data

    def parse_legal_entity(self, raw_data: Dict) -> Dict[str, Any]:
        """
        Parse raw JSON data into structured legal entity format
```

```
        """
        entity = {
            'registration_number': raw_data.get(' '),
            'name': raw_data.get(' '),
            'registration_date': raw_data.get(' '),
            'legal_form': raw_data.get(''),
            'entity_type': raw_data.get(''),
            'address': raw_data.get('  '),
            'shareholders': [],
            'authorized_representatives': [],
            'beneficial_owners': [],
            'business_activities': [],
            'restructuring_events': []
        }

        # Parse shareholders/members
        shareholders_data = raw_data.get(' ,  ', [])
        if isinstance(shareholders_data, list):
            for sh in shareholders_data:
                entity['shareholders'].append({
                    'classification': sh.get(''),
                    'country': sh.get(' '),
                    'patronymic': sh.get('//- '),
                    'name': sh.get(''),
                    'gender': sh.get(''),
                    'registration_date': sh.get(' ')
                })

        # Parse authorized representatives
        auth_rep_data = raw_data.get(
            '     ,    ',
            []
        )
        if isinstance(auth_rep_data, list):
            for rep in auth_rep_data:
                entity['authorized_representatives'].append({
                    'position': rep.get(' '),
                    'country': rep.get(' '),
                    'patronymic': rep.get('//- '),
                    'name': rep.get(''),
                    'gender': rep.get(''),
                    'registration_date': rep.get(' ')
                })

        # Parse beneficial owners
        bo_data = raw_data.get(' ', [])
        if isinstance(bo_data, list):
```

```
            for bo in bo_data:
                entity['beneficial_owners'].append({
                    'classification': bo.get(''),
                    'patronymic': bo.get('//- '),
                    'name': bo.get(''),
                    'gender': bo.get(''),
                    'registration_date': bo.get(' ')
                })

        # Parse business activities
        activity_data = raw_data.get('   ', [])
        if isinstance(activity_data, list):
            for act in activity_data:
                entity['business_activities'].append({
                    'status': act.get(''),
                    'registration_date': act.get(' '),
                    'activity_direction': act.get('  ')
                })

        # Parse restructuring events
        restructure_data = raw_data.get('    ', [])
        if isinstance(restructure_data, list):
            for rst in restructure_data:
                entity['restructuring_events'].append({
                    'restructuring_type': rst.get('   '),
                    'registration_date': rst.get(' '),
                    'previous_name': rst.get('   '),
                    'change_notes': rst.get(' ')
                })

        return entity

    def save_to_json(self, data: List[Dict], filename: str = None) -> str:
        """Save extracted data to JSON file"""
        if filename is None:
            timestamp = datetime.now().strftime('%Y%m%d_%H%M%S')
            filename = f"legal_entities_{timestamp}.json"

        filepath = os.path.join(self.output_dir, filename)

        with open(filepath, 'w', encoding='utf-8') as f:
            json.dump(data, f, ensure_ascii=False, indent=2)

        print(f"Data saved to: {filepath}")
        return filepath

    def extract_all(self) -> List[Dict]:
```

```
        """
        Main extraction method - downloads and parses all data
        """
        print("Starting data extraction from Mongolia Open Data Portal...")

        # Download raw data
        raw_data = self.download_json_data()

        if raw_data is None:
            print("Attempting paginated download...")
            raw_data = self.download_with_pagination()

        if not raw_data:
            print("No data retrieved")
            return []

        # Parse entities
        parsed_entities = []
        for item in raw_data:
            parsed_entity = self.parse_legal_entity(item)
            parsed_entities.append(parsed_entity)

        # Save to file
        self.save_to_json(parsed_entities)

        print(f"Extraction complete. Total entities: {len(parsed_entities)}")
        return parsed_entities


# Main execution
if __name__ == "__main__":
    extractor = MongoliaOpenDataExtractor(output_dir="extracted_data")
    entities = extractor.extract_all()

    # Print sample
    if entities:
        print("\nSample entity:")
        print(json.dumps(entities[0], indent=2, ensure_ascii=False))
```

Listing 1: JSON Data Extraction from Mongolia Open Data Portal

**6.2.4.2 Web Scraping from Mongolian Stock Exchange** The following Python code scrapes company information from the Mongolian Stock Exchange website:

```
import requests
from bs4 import BeautifulSoup
import json
```

```python
import time
import os
from datetime import datetime
from typing import Optional, List, Dict, Any
from urllib.parse import urljoin
import re

class MSEScraper:
    """
    Web Scraper for Mongolian Stock Exchange (MSE)
    Source: https://mse.mn/mn
    """

    def __init__(self, output_dir: str = "data"):
        self.base_url = "https://mse.mn"
        self.language = "mn"  # Mongolian language
        self.output_dir = output_dir
        self.session = requests.Session()
        self.session.headers.update({
            'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36',
            'Accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8',
            'Accept-Language': 'mn-MN,mn;q=0.9,en-US;q=0.8,en;q=0.7',
            'Connection': 'keep-alive'
        })

        # Rate limiting
        self.request_delay = 2  # seconds between requests

        # Create output directory
        if not os.path.exists(output_dir):
            os.makedirs(output_dir)

    def _make_request(self, url: str, params: Dict = None) -> Optional[BeautifulSoup]:
        """Make HTTP request with rate limiting and error handling"""
        try:
            time.sleep(self.request_delay)
            response = self.session.get(url, params=params, timeout=30)
            response.raise_for_status()
            response.encoding = 'utf-8'
            return BeautifulSoup(response.text, 'html.parser')
        except requests.RequestException as e:
            print(f"Request error for {url}: {e}")
            return None

```

```python
    def get_listed_companies(self) -> List[Dict[str, Any]]:
        """
        Scrape list of all companies listed on MSE
        """
        companies = []

        # Main securities listing page
        listing_url = f"{self.base_url}/{self.language}/security"

        print(f"Fetching listed companies from: {listing_url}")
        soup = self._make_request(listing_url)

        if not soup:
            return companies

        # Find company listing table
        # Note: Actual selectors depend on MSE website structure
        company_tables = soup.find_all('table', class_='table')

        for table in company_tables:
            rows = table.find_all('tr')[1:]  # Skip header

            for row in rows:
                cols = row.find_all('td')
                if len(cols) >= 3:
                    company = {
                        'symbol': cols[0].get_text(strip=True),
                        'name': cols[1].get_text(strip=True),
                        'sector': cols[2].get_text(strip=True) if len(cols) > 2 else None,
                        'detail_url': None
                    }

                    # Extract detail link
                    link = cols[1].find('a')
                    if link and link.get('href'):
                        company['detail_url'] = urljoin(self.base_url, link['href'])

                    companies.append(company)

        print(f"Found {len(companies)} listed companies")
        return companies

    def get_company_details(self, company_url: str) -> Optional[Dict[str, Any]]:
        """
        Scrape detailed information for a specific company
```

```
        """
        if not company_url:
            return None

        print(f"Fetching company details from: {company_url}")
        soup = self._make_request(company_url)

        if not soup:
            return None

        details = {
            'url': company_url,
            'scraped_at': datetime.now().isoformat()
        }

        # Company profile section
        profile_section = soup.find('div', class_='company-profile')
        if profile_section:
            details['profile'] = self._parse_profile(profile_section)

        # Trading information
        trading_section = soup.find('div', class_='trading-info')
        if trading_section:
            details['trading'] = self._parse_trading_info(trading_section)

        # Company information table
        info_table = soup.find('table', class_='company-info')
        if info_table:
            details['company_info'] = self._parse_info_table(info_table)

        # Shareholders information
        shareholders_section = soup.find('div', {'id': 'shareholders'})
        if shareholders_section:
            details['major_shareholders'] = self._parse_shareholders(
    shareholders_section)

        # Board members / Management
        management_section = soup.find('div', {'id': 'management'})
        if management_section:
            details['management'] = self._parse_management(management_section)

        # Financial highlights
        financial_section = soup.find('div', {'id': 'financials'})
        if financial_section:
            details['financials'] = self._parse_financials(financial_section)

        return details
```

```python

    def _parse_profile(self, section) -> Dict[str, str]:
        """Parse company profile section"""
        profile = {}

        # Company name
        name_elem = section.find(['h1', 'h2', 'h3'], class_='company-name')
        if name_elem:
            profile['name'] = name_elem.get_text(strip=True)

        # Description
        desc_elem = section.find('p', class_='description')
        if desc_elem:
            profile['description'] = desc_elem.get_text(strip=True)

        # Extract key-value pairs
        info_items = section.find_all(['div', 'li'], class_='info-item')
        for item in info_items:
            label = item.find(['span', 'strong'], class_='label')
            value = item.find(['span', 'div'], class_='value')
            if label and value:
                key = label.get_text(strip=True).rstrip(':')
                profile[key] = value.get_text(strip=True)

        return profile

    def _parse_trading_info(self, section) -> Dict[str, Any]:
        """Parse trading information"""
        trading = {}

        # Current price
        price_elem = section.find('span', class_='current-price')
        if price_elem:
            price_text = price_elem.get_text(strip=True)
            trading['current_price'] = self._extract_number(price_text)

        # Price change
        change_elem = section.find('span', class_='price-change')
        if change_elem:
            trading['price_change'] = change_elem.get_text(strip=True)

        # Volume
        volume_elem = section.find('span', class_='volume')
        if volume_elem:
            trading['volume'] = self._extract_number(volume_elem.get_text(strip=
    True))

```

```
        # Market cap
        mcap_elem = section.find('span', class_='market-cap')
        if mcap_elem:
            trading['market_cap'] = mcap_elem.get_text(strip=True)

        return trading

    def _parse_info_table(self, table) -> Dict[str, str]:
        """Parse information table"""
        info = {}
        rows = table.find_all('tr')

        for row in rows:
            cells = row.find_all(['th', 'td'])
            if len(cells) >= 2:
                key = cells[0].get_text(strip=True).rstrip(':')
                value = cells[1].get_text(strip=True)
                info[key] = value

        return info

    def _parse_shareholders(self, section) -> List[Dict[str, Any]]:
        """Parse major shareholders information"""
        shareholders = []

        table = section.find('table')
        if table:
            rows = table.find_all('tr')[1:]  # Skip header

            for row in rows:
                cols = row.find_all('td')
                if len(cols) >= 2:
                    shareholder = {
                        'name': cols[0].get_text(strip=True),
                        'shares': self._extract_number(cols[1].get_text(strip=True)
) if len(cols) > 1 else None,
                        'percentage': cols[2].get_text(strip=True) if len(cols) > 2
 else None
                    }
                    shareholders.append(shareholder)

        return shareholders

    def _parse_management(self, section) -> List[Dict[str, str]]:
        """Parse management/board members information"""
        management = []

```

```
        members = section.find_all(['div', 'li'], class_='member')
        for member in members:
            name_elem = member.find(['h4', 'span'], class_='name')
            position_elem = member.find(['p', 'span'], class_='position')

            if name_elem:
                management.append({
                    'name': name_elem.get_text(strip=True),
                    'position': position_elem.get_text(strip=True) if position_elem
    else None
                })

        # Alternative: table format
        table = section.find('table')
        if table and not management:
            rows = table.find_all('tr')[1:]
            for row in rows:
                cols = row.find_all('td')
                if len(cols) >= 2:
                    management.append({
                        'name': cols[0].get_text(strip=True),
                        'position': cols[1].get_text(strip=True)
                    })

        return management

    def _parse_financials(self, section) -> Dict[str, Any]:
        """Parse financial highlights"""
        financials = {}

        table = section.find('table')
        if table:
            rows = table.find_all('tr')
            for row in rows:
                cells = row.find_all(['th', 'td'])
                if len(cells) >= 2:
                    metric = cells[0].get_text(strip=True)
                    value = cells[1].get_text(strip=True)
                    financials[metric] = value

        return financials

    def _extract_number(self, text: str) -> Optional[float]:
        """Extract numeric value from text"""
        if not text:
            return None
        # Remove non-numeric characters except decimal point
```

```
        cleaned = re.sub(r'[^\d.]', '', text.replace(',', ''))
        try:
            return float(cleaned) if cleaned else None
        except ValueError:
            return None

    def get_market_indices(self) -> List[Dict[str, Any]]:
        """Scrape market indices data"""
        indices = []
        index_url = f"{self.base_url}/{self.language}/index"

        soup = self._make_request(index_url)
        if not soup:
            return indices

        index_cards = soup.find_all('div', class_='index-card')
        for card in index_cards:
            name_elem = card.find(['h3', 'h4'], class_='index-name')
            value_elem = card.find('span', class_='index-value')
            change_elem = card.find('span', class_='index-change')

            if name_elem and value_elem:
                indices.append({
                    'name': name_elem.get_text(strip=True),
                    'value': self._extract_number(value_elem.get_text(strip=True)),
                    'change': change_elem.get_text(strip=True) if change_elem else
None
                })

        return indices

    def scrape_all(self) -> Dict[str, Any]:
        """
        Main scraping method - collects all available data
        """
        print("Starting MSE web scraping...")

        result = {
            'scraped_at': datetime.now().isoformat(),
            'source': self.base_url,
            'indices': [],
            'companies': []
        }

        # Get market indices
        print("\n1. Scraping market indices...")
        result['indices'] = self.get_market_indices()
```

```
322
323         # Get listed companies
324         print("\n2. Scraping listed companies...")
325         companies = self.get_listed_companies()
326
327         # Get details for each company
328         print("\n3. Scraping individual company details...")
329         for i, company in enumerate(companies):
330             print(f"Processing {i+1}/{len(companies)}: {company.get('name', '
    Unknown')}")
331
332             if company.get('detail_url'):
333                 details = self.get_company_details(company['detail_url'])
334                 if details:
335                     company.update(details)
336
337             result['companies'].append(company)
338
339         # Save results
340         self.save_to_json(result)
341
342         print(f"\nScraping complete. Total companies: {len(result['companies'])}")
343         return result
344
345     def save_to_json(self, data: Dict, filename: str = None) -> str:
346         """Save scraped data to JSON file"""
347         if filename is None:
348             timestamp = datetime.now().strftime('%Y%m%d_%H%M%S')
349             filename = f"mse_data_{timestamp}.json"
350
351         filepath = os.path.join(self.output_dir, filename)
352
353         with open(filepath, 'w', encoding='utf-8') as f:
354             json.dump(data, f, ensure_ascii=False, indent=2)
355
356         print(f"Data saved to: {filepath}")
357         return filepath
358
359
360 # Main execution
361 if __name__ == "__main__":
362     scraper = MSEScraper(output_dir="mse_extracted_data")
363
364     # Scrape all data
365     data = scraper.scrape_all()
366
367     # Print summary
```

```
    print("\n" + "="*50)
    print("SCRAPING SUMMARY")
    print("="*50)
    print(f"Market Indices: {len(data.get('indices', []))}")
    print(f"Listed Companies: {len(data.get('companies', []))}")

    # Print sample company
    if data.get('companies'):
        print("\nSample company data:")
        print(json.dumps(data['companies'][0], indent=2, ensure_ascii=False))
```

Listing 2: Web Scraping from Mongolian Stock Exchange

#### 6.2.4.3 Data Integration Script

The following script integrates data from both sources:

```
import json
from typing import Dict, List, Any, Optional
from datetime import datetime

class DataIntegrator:
    """
    Integrates data from Mongolia Open Data Portal and MSE
    Links legal entity data with stock exchange information
    """

    def __init__(self):
        self.legal_entities = []
        self.mse_companies = []
        self.integrated_data = []

    def load_legal_entities(self, filepath: str) -> None:
        """Load legal entities data from JSON file"""
        with open(filepath, 'r', encoding='utf-8') as f:
            self.legal_entities = json.load(f)
        print(f"Loaded {len(self.legal_entities)} legal entities")

    def load_mse_data(self, filepath: str) -> None:
        """Load MSE data from JSON file"""
        with open(filepath, 'r', encoding='utf-8') as f:
            data = json.load(f)
            self.mse_companies = data.get('companies', [])
        print(f"Loaded {len(self.mse_companies)} MSE companies")

    def find_matching_entity(self, mse_company: Dict) -> Optional[Dict]:
        """
        Find matching legal entity for MSE company
        Uses name matching with normalization
```

```
        """
        mse_name = mse_company.get('name', '').strip().upper()

        for entity in self.legal_entities:
            entity_name = entity.get('name', '').strip().upper()

            # Exact match
            if entity_name == mse_name:
                return entity

            # Partial match (company name contained)
            if mse_name in entity_name or entity_name in mse_name:
                return entity

        return None

    def integrate(self) -> List[Dict[str, Any]]:
        """
        Integrate data from both sources
        """
        print("Starting data integration...")

        for mse_company in self.mse_companies:
            integrated_record = {
                'mse_data': mse_company,
                'legal_entity_data': None,
                'integration_status': 'no_match',
                'integrated_at': datetime.now().isoformat()
            }

            # Find matching legal entity
            matching_entity = self.find_matching_entity(mse_company)

            if matching_entity:
                integrated_record['legal_entity_data'] = matching_entity
                integrated_record['integration_status'] = 'matched'

            self.integrated_data.append(integrated_record)

        # Statistics
        matched = sum(1 for r in self.integrated_data
                      if r['integration_status'] == 'matched')
        print(f"Integration complete: {matched}/{len(self.integrated_data)} matched
    ")

        return self.integrated_data

```

```
    def save_integrated_data(self, filepath: str) -> None:
        """Save integrated data to JSON file"""
        with open(filepath, 'w', encoding='utf-8') as f:
            json.dump(self.integrated_data, f, ensure_ascii=False, indent=2)
        print(f"Integrated data saved to: {filepath}")


# Usage example
if __name__ == "__main__":
    integrator = DataIntegrator()
    integrator.load_legal_entities("extracted_data/legal_entities.json")
    integrator.load_mse_data("mse_extracted_data/mse_data.json")
    integrated = integrator.integrate()
    integrator.save_integrated_data("integrated_data/combined_data.json")
```

Listing 3: Data Integration Script

#### 6.2.5 Resources Classification

### 6.3 Design Decisions and Rationale

#### 6.3.1 Choice of Data Sources

**Decision:** Use Mongolia Open Data Portal as primary source with MSE as supplementary source.

**Strengths:**

- Official government data ensures reliability and legal validity
- Direct JSON download eliminates complex parsing requirements
- Open data license permits unrestricted use for the project
- MSE data enriches information for publicly traded companies

**Weaknesses:**

- Government data may have update delays
- MSE website structure may change, requiring scraper maintenance
- No direct linkage keys between the two data sources

#### 6.3.2 Choice of Extraction Methods

**Decision:** Direct API/download for Open Data Portal; web scraping for MSE.

**Rationale:** The Open Data Portal provides structured JSON access, making direct download the most efficient approach. MSE lacks a public API, necessitating web scraping with appropriate rate limiting to avoid server overload.

#### 6.3.3 Data Integration Strategy

**Decision:** Name-based matching for linking MSE companies to legal entities.

**Strengths:**

- Simple implementation without requiring additional external data
- Effective for well-known listed companies with consistent naming

**Weaknesses:**

- May miss matches due to name variations
- Cannot automatically resolve ambiguous matches
- Manual verification recommended for critical linkages

Table 10: Data Resources Classification

| **Data Resource** | **Source** | **Classification** | **Justification** |
|---|---|---|---|
| Legal Entity Basic Info | Open Data Portal | Core | Essential identification data for all entities |
| Shareholder Information | Open Data Portal | Core | Central to ownership visualization purpose |
| Authorized Representatives | Open Data Portal | Core | Key governance and representation data |
| Ultimate Beneficial Owners | Open Data Portal | Core | Critical for transparency objectives |
| Business Activities | Open Data Portal | Contextual | Supporting information for entity characterization |
| Restructuring History | Open Data Portal | Contextual | Historical context for entity changes |
| Stock Trading Data | MSE | Contextual | Supplementary market information |
| Company Financial Data | MSE | Contextual | Additional context for listed companies |