Phase 1 Submission

State Alchemist Research and Mission Management System

Team Name: Agent746 **Team Number:** 13

Team Members: Eashan Thakur, Hemang Joshi, Shardul Joshi

1. Idea Formulation & Context

1.1 Mini-World Title

State Alchemist Research and Mission Management System
Mini-world: Fullmetal Alchemist Universe

1.2 Context and Narrative Integration

Alchemy is a formalized science in this universe. Licensed State Alchemists perform missions, research labs conduct experiments, and military contracts involve investigating unnatural phenomena. State Alchemists coordinate with military units for certain field missions; research labs use certified alchemical devices to standardize experiments. This database tracks:

- State Alchemists (qualifications, school, rank), their missions, and research projects.
- Incidents (anomalies, transmutations), artifacts recovered, and homunculi case files.
- Alchemical devices, ingredients, and secure research management.
- Mission logs to detect timeline-impacting anomalies or illegal transmutations.
- Military unit participation and assigned command structures.
- Inventory of registered AlchemicalDevices and device usage in experiments.

1.3 User Identification & Interaction

Primary Users:

- **State Alchemist Officers:** Manage missions, submit reports, request supplies.
- **Research Lab Managers:** Track experiments, request ingredients, store artifact data.
- **Military Mission Command:** Assign alchemists, update mission status, coordinate with military units.
- **Forensics Analysts:** Log homunculi and artifacts examination.
- **Civilian Intake Officers:** Record civilian reports and assign investigations.
- **Research Lab Directors:** Oversee experimental procedures, device calibration, and safety protocols.

- **Military Command:** Coordinate unit deployments and joint operations with State Alchemists.

1.4 Purpose of the Database

- **Consistency:** Prevent duplication, maintain structured records.
- **Referential integrity:** Ensure missions, logs, and assignments are linked correctly.
- **Analytics:** Enable cross-entity queries for reports (e.g., mission success rate by school).
- **Security & audit:** Control access to dangerous materials and track changes; support secure storage policies for high-danger artifacts, and enable audits for illegal human transmutation attempts.

2. Database Requirements

2.1 Entity Types

1. **Alchemist**
   - **Primary key:** AlchemistID
   - **Attributes:** Name (First, Middle, Last) [composite], Rank, School, LicenseNumber+NationCode [two-key], RankHistory [multivalued - normalized as AlchemistRankHistory table], Specializations [multivalued - normalized as AlchemistSpecialization table]
2. **ResearchLab**
   - **Primary key:** LabID
   - **Attributes:** LabName, Location (City, Region) [composite], SecurityLevel
3. **Mission**
   - **Primary key:** MissionID
   - **Attributes:** MissionCode+Year [two-key], MissionType, StartDate, EndDate, Status
4. **Artifact**
   - **Primary key:** ArtifactID
   - **Attributes:** Name, CreationDate, KnownHistory, DangerLevel, CurrentLocation (LabID or Archive)
5. **Homunculus**
   - **Primary key:** HomunculusID
   - **Attributes:** GivenName, CreationMethod, ActiveStatus, RegenerationIndex, LabID [FK, nullable - indicates custody/containment]
6. **Ingredient**
   - **Primary key:** IngredientID
   - **Attributes:** ChemicalName, StockQuantity, HazardClass

7. **MilitaryUnit**
   - **Primary key:** UnitID
   - **Attributes:** UnitName, BaseLocation (City, Region) [composite], CommanderID [FK to Alchemist, nullable]
   - **Purpose:** Models Amestrian military divisions that coordinate missions and provide logistics.
8. **AlchemicalDevice**
   - **Primary key:** DeviceID
   - **Attributes:** DeviceName, DeviceType (TransmutationCircleAdapter, Stabilizer, etc.), LabID [FK to ResearchLab]
   - **Purpose:** Track devices used in transmutations; ties into experiments for standardization and audit.

2.2 Weak Entity Types

1. **MissionLog** (depends on Mission)
   - **Owner key:** MissionID
   - **Partial key:** LogSeqNo
   - **Composite key:** (MissionID, LogSeqNo)
   - **Attributes:** Timestamp, Description, Outcome, ReporterID [FK to Alchemist], EvidenceRef [FK to Artifact, nullable]
2. **TransmutationAttempt** (depends on Artifact)
   - **Owner key:** ArtifactID
   - **Partial key:** AttemptNo
   - **Composite key:** (ArtifactID, AttemptNo)
   - **Attributes:** AttemptDate, PerformerID [FK to Alchemist], Result, Casualties [derived], Notes, EvidenceRef [FK to Artifact, nullable]

2.3 Two Entities with Two-Key Attributes

- **Alchemist:** (LicenseNumber, NationCode) - uniquely identifies each State Alchemist across different nations
- **Mission:** (MissionCode, Year) - mission codes can repeat across years

2.4 Relationships (with Cardinalities and Constraints)

1. **AssignedTo:** Alchemist — (M:N) — Mission *Mission: total, Alchemist: optional*
2. **Conducts:** ResearchLab — (1:N) — TransmutationAttempt *TransmutationAttempt: total participation*
3. **Contains:** ResearchLab — (1:N) — Ingredient *Ingredient: optional*
4. **RecoveredFrom:** Artifact — (1:N) — MissionLog *Links artifacts to the mission logs where they were discovered*

5. **Creates:** Alchemist — (1:N) — TransmutationAttempt *TransmutationAttempt: total*
6. **Examines:** ForensicsAnalyst — (M:N) — Artifact *Zero or more participation both sides*
7. **ResearchLabContainsHomunculus:** ResearchLab — (1:N) — Homunculus *Cardinality: ResearchLab (1) : (0..N) Homunculus. Homunculus typically housed in a lab or free (0..1 lab at a time). Tracks lab custody of captured homunculi.*
8. **RegistersDevice:** ResearchLab — (1:N) — AlchemicalDevice *Each device registered to one lab; labs can have multiple devices*
9. **Commands:** Alchemist — (0..1:N) — MilitaryUnit *A unit may have one Alchemist commander; an Alchemist may command zero or one unit*

Multi-Entity Relationships (arity n = 4)

1. **UseOfIngredientInExperiment** (4-ary relationship): TransmutationAttempt × Ingredient × Alchemist × AlchemicalDevice
   - **Participating Entities:** TransmutationAttempt, Ingredient, Alchemist, AlchemicalDevice
   - **Attributes:** Quantity, IngredientRole (Catalyst, Solvent, Reagent), MeasurementUnit (g, ml, etc.)
   - **Constraints:** TransmutationAttempt must belong to 1 Artifact; Alchemist must be creator or performer; Device and Attempt must be from the same lab
   - **Rationale:** ResearchLab removed as it's redundant since both Device and Attempt link to a lab
2. **MissionAssignmentRole** (4-ary relationship): Mission × Alchemist × MilitaryUnit × ResearchLab
   - **Participating Entities:** Mission, Alchemist, MilitaryUnit, ResearchLab
   - **Attributes:** Role (domain: {Lead, Support, Medic, Scout}), AssignmentDate
   - **Purpose:** Stores the alchemist's role, coordinating military unit, and supporting research lab for missions
   - **Constraints:** Each combination of Mission, Alchemist, MilitaryUnit, and ResearchLab is unique; Role describes the alchemist's function
   - **Rationale:** Artifact removed; artifact focus can appear via MissionLog or artifact recovery relations. ResearchLab added to track which lab provides research support for the mission
   - **Note:** This supersedes simple AssignedTo; AssignedTo remains for basic mission assignments while MissionAssignmentRole captures detailed role/unit/lab coordination information

2.5 Subclass Hierarchy

- **Alchemist** (superclass)
  - **ForensicsAnalyst:** CertificationLevel, LabAffiliation
  - **FieldOperative:** CombatRating, FieldExperienceYears
- Single inheritance: an Alchemist may belong to one subclass or none
- Disjoint specialization: an Alchemist cannot be both ForensicsAnalyst and FieldOperative simultaneously

2.6 Composite, Multi-Valued, and Derived Attributes

- **Composite:**
  - Alchemist.Name = (First, Middle, Last)
  - ResearchLab.Location = (City, Region)
  - MilitaryUnit.BaseLocation = (City, Region)
- **Multi-valued:**
  - Alchemist.Specializations (normalized as AlchemistSpecialization table)
  - Alchemist.RankHistory (normalized as AlchemistRankHistory(AlchemistID, Rank, FromDate, ToDate) table)
- **Derived:**
  - TransmutationAttempt.Casualties (derived from outcome records)

2.7 Domain and Integrity Constraints

- **Date format:** YYYY-MM-DD. EndDate $\geq$ StartDate for missions & attempts.
- **DangerLevel domain:** {Low, Medium, High, Critical}. Philosopher's Stones $\rightarrow$ Critical.
- **Non-negative values:** StockQuantity ≥ 0, RegenerationIndex ≥ 0
- **No human transmutation:** Application-level constraint or DB CHECK that forbids TransmutationAttempt on an Artifact flagged IsHuman = TRUE
- **Referential integrity:** All PerformerID, ReporterID, CommanderID must reference existing Alchemist.AlchemistID
- **Security-level storage:** Artifact.DangerLevel = Critical must have CurrentLocation in a ResearchLab with SecurityLevel $\in$ {High, Critical}
- **Role enumerations:** Role $\in$ {Lead, Support, Medic, Scout}; IngredientRole $\in$ {Catalyst, Solvent, Reagent}
- **HazardClass semantics:** Ingredient.HazardClass $\in$ {1, 2, 3, 4} where higher values indicate greater hazard; Class 4 ingredients require SecurityLevel = High or Critical

2.8 Key Table Definitions

New Entity Tables

```
CREATE TABLE MilitaryUnit(
    UnitID INT PRIMARY KEY,
    UnitName VARCHAR(200) NOT NULL,
    City VARCHAR(100),
    Region VARCHAR(100),
    CommanderID INT
);
CREATE TABLE AlchemicalDevice(
    DeviceID INT PRIMARY KEY,
    DeviceName VARCHAR(200) NOT NULL,
    DeviceType VARCHAR(100),
    LabID INT
);
```

Normalization Tables (Multi-valued Attributes)

```
CREATE TABLE AlchemistRankHistory(
    AlchemistID INT,
    Rank VARCHAR(50),
    FromDate DATE,
    ToDate DATE,
    PRIMARY KEY (AlchemistID, FromDate)
);
CREATE TABLE AlchemistSpecialization(
    AlchemistID INT,
    Specialization VARCHAR(100),
    PRIMARY KEY (AlchemistID, Specialization)
);
```

Multi-Entity Relationship Tables

```
CREATE TABLE UseOfIngredientInExperiment(
    AttemptArtifactID INT,
    AttemptNo INT,
    IngredientID INT,
    AlchemistID INT,
    DeviceID INT,
    Quantity DECIMAL(10,2) NOT NULL,
```

```
    IngredientRole VARCHAR(20),
    MeasurementUnit VARCHAR(10),
    PRIMARY KEY (AttemptArtifactID, AttemptNo, IngredientID,
                 AlchemistID, DeviceID)
);
CREATE TABLE MissionAssignmentRole(
    MissionID INT,
    AlchemistID INT,
    UnitID INT,
    LabID INT,
    Role VARCHAR(20),
    AssignmentDate DATE,
    PRIMARY KEY (MissionID, AlchemistID, UnitID, LabID)
);
```

---

3. Functional Requirements

3.1 Retrieval Operations

3.1.1 Selection

Query: Retrieve all active Extermination missions that started after January 1, 2024. **Description:** Filter the Mission table to return only records where MissionType is 'Extermination', StartDate is after 2024-01-01, and Status is 'Active'. This helps mission command identify ongoing critical operations.

3.1.2 Projection

Query: Display the name, rank, and combat rating of all FieldOperatives. **Description:** Join the Alchemist and FieldOperative tables, selecting only the relevant attributes (First Name, Last Name, Rank, CombatRating). This provides a focused view for personnel evaluation without exposing unnecessary personal information.

3.1.3 Aggregation

Query: Calculate the average RegenerationIndex of all active Homunculi. **Description:** Compute AVG(RegenerationIndex) for Homunculus records where ActiveStatus is 'True'. This metric helps researchers assess the current threat level of contained homunculi and prioritize containment resources.

3.1.4 Search

Query: Find all artifacts whose name contains the term "philosopher" (case-insensitive). **Description:** Perform a partial text match on the Artifact.Name field using LIKE op-

erator with wildcards. This enables researchers to quickly locate Philosopher's Stones or related artifacts even with incomplete naming information.

3.2 Analysis Reports

Report 1: Monthly mission success rate by Alchemist School

```
SELECT A.School,
  COUNT(DISTINCT M.MissionID) AS TotalMissions,
  SUM(CASE WHEN ML.Outcome='Success' THEN 1 ELSE 0 END) AS Successes,
  ROUND(100.0*SUM(CASE WHEN ML.Outcome='Success' THEN 1 ELSE 0 END)
    /COUNT(DISTINCT M.MissionID),2) AS SuccessRate
FROM Mission M
JOIN AssignedTo AT ON M.MissionID = AT.MissionID
JOIN Alchemist A ON AT.AlchemistID = A.AlchemistID
JOIN MissionLog ML ON M.MissionID = ML.MissionID
WHERE M.StartDate BETWEEN '2025-01-01' AND '2025-01-31'
GROUP BY A.School;
```

Report 2: Top 5 ingredients by total stock across labs

```
SELECT I.ChemicalName, L.LabName, SUM(I.StockQuantity) AS TotalStock
FROM Ingredient I
JOIN ResearchLab L ON I.LabID = L.LabID
GROUP BY I.ChemicalName, L.LabName
ORDER BY TotalStock DESC
LIMIT 5;
```

Report 3: Homunculi with high regeneration index in custody

```
SELECT H.HomunculusID, H.GivenName, H.RegenerationIndex,
       RL.LabName, RL.SecurityLevel
FROM Homunculus H
LEFT JOIN ResearchLab RL ON H.LabID = RL.LabID
WHERE H.RegenerationIndex > 50
ORDER BY H.RegenerationIndex DESC;
```

3.3 Modification Operations

3.3.1 Insertion (with automatic constraint checking)

Insert a new mission with date validation:

```sql
-- Constraint ensures EndDate >= StartDate
ALTER TABLE Mission
ADD CONSTRAINT chk_mission_dates
CHECK (EndDate IS NULL OR EndDate >= StartDate);
-- This insertion succeeds
INSERT INTO Mission(MissionID, MissionCode, Year, MissionType,
                    StartDate, EndDate, Status)
VALUES (1001, 'EXTR-09', 2025, 'Extermination',
        '2025-02-20', '2025-02-25', 'Active');
-- This would fail due to constraint violation
INSERT INTO Mission(MissionID, MissionCode, Year, MissionType,
                    StartDate, EndDate, Status)
VALUES (1002, 'INV-10', 2025, 'Investigation',
        '2025-03-15', '2025-03-10', 'Active');
-- ERROR: EndDate < StartDate violates constraint
```

3.3.2 Update (with cascading business rules)

Update mission status with automatic FieldOperative rating adjustment:

```sql
-- When a mission is closed as failure, decrease combat ratings
-- Trigger: AFTER UPDATE ON Mission
-- Business Rule: Failed missions reduce FieldOperative CombatRating by 1
UPDATE Mission
SET Status = 'Closed', Outcome = 'Failure'
WHERE MissionID = 1001;
-- Automatically triggers cascade update:
UPDATE FieldOperative
SET CombatRating = GREATEST(0, CombatRating - 1)
WHERE AlchemistID IN (
    SELECT AlchemistID FROM AssignedTo
    WHERE MissionID = 1001
);
```

3.3.3 Deletion (maintaining referential integrity)

Delete mission with referential integrity protection:

```sql
-- Foreign key with CASCADE deletes associated mission logs
ALTER TABLE MissionLog
ADD FOREIGN KEY (MissionID) REFERENCES Mission(MissionID)
ON DELETE CASCADE;
```

```sql
-- Before deletion, check for critical evidence
-- Trigger: BEFORE DELETE ON Mission
-- Prevents deletion if critical evidence exists
-- This deletion succeeds (no critical evidence, logs cascade delete)
DELETE FROM Mission WHERE MissionID = 1005;
-- This deletion may fail if mission has critical evidence linked
DELETE FROM Mission WHERE MissionID = 1001;
-- ERROR: Cannot delete mission with critical evidence references
```
