Soma Swahili — ERD & UML

This canvas contains a draft ERD, detailed table definitions (attributes, PKs, FKs), UML class diagrams and a couple of sequence diagrams for common flows (signup, booking a lesson). Use this as the living design — we can iterate and I can produce SQL DDL or visual exports when you say "export".

---

1) Design goals / assumptions

- Two main user roles: **Learner (Student)** and **Tutor**. Admin is separate role for platform maintenance.
- Tutors publish **Courses** or **Lessons** (can be single sessions or multi-session courses).
- Booking system: students book a **Session** (scheduled lesson) with a tutor. Payments may be required.
- Messaging between student and tutor (in-app chat). Notifications for key events.
- Ratings & Reviews for tutors/sessions.
- Support for uploading learning resources (files/links).
- Keep the schema normalized (3NF) while pragmatic for queries (add indexes where needed).

---

2) ERD (Mermaid format)

```
erDiagram
    USERS {
        int id PK
        varchar email
        varchar password_hash
        varchar role
        varchar display_name
        datetime created_at
        datetime updated_at
        boolean verified
    }

    PROFILES {
        int id PK
        int user_id FK
        varchar bio
        varchar location
        varchar language_native
        varchar language_learning
        varchar avatar_url
        int rating_avg
        int rating_count
```

```
}

TUTORS {
    int id PK
    int user_id FK
    boolean approved
    text qualifications
    text specialties
}

COURSES {
    int id PK
    int tutor_id FK
    varchar title
    text description
    decimal price_per_session
    varchar currency
    boolean published
    datetime created_at
}

SESSIONS {
    int id PK
    int course_id FK
    int tutor_id FK
    int student_id FK
    datetime start_time
    datetime end_time
    enum status
    boolean paid
    datetime booked_at
}

PAYMENTS {
    int id PK
    int session_id FK
    int payer_id FK
    decimal amount
    varchar currency
    varchar provider
    varchar provider_txn_id
    enum status
    datetime created_at
}

MESSAGES {
    int id PK
    int from_user_id FK
```

```
        int to_user_id FK
        int session_id FK NULL
        text body
        datetime sent_at
        boolean read
    }

    REVIEWS {
        int id PK
        int session_id FK NULL
        int reviewer_id FK
        int reviewee_id FK
        int rating
        text comment
        datetime created_at
    }

    RESOURCES {
        int id PK
        int course_id FK NULL
        int uploaded_by FK
        varchar title
        varchar url
        varchar type
        datetime uploaded_at
    }

    AVAILABILITIES {
        int id PK
        int tutor_id FK
        date weekday
        time start_time
        time end_time
        varchar timezone
    }

    NOTIFICATIONS {
        int id PK
        int user_id FK
        varchar type
        text payload
        boolean read
        datetime created_at
    }

    USERS ||--o{ PROFILES : has
    USERS ||--o{ TUTORS : may_be
    TUTORS ||--o{ COURSES : creates
    COURSES ||--o{ SESSIONS : schedule
    USERS ||--o{ SESSIONS : books
    SESSIONS ||--o{ PAYMENTS : has
    USERS ||--o{ MESSAGES : sends
    SESSIONS ||--o{ MESSAGES : context_for
    USERS ||--o{ REVIEWS : writes
    SESSIONS ||--o{ REVIEWS : may_produce
    COURSES ||--o{ RESOURCES : contains
    TUTORS ||--o{ AVAILABILITIES : defines
    USERS ||--o{ NOTIFICATIONS : receives
```

---

3) Tables & attributes (detailed)

Below each table I list key fields, constraints and suggested indexes.

users

- id BIGINT PRIMARY KEY AUTO_INCREMENT
- email VARCHAR(255) UNIQUE NOT NULL
- password_hash VARCHAR(255) NOT NULL
- role ENUM('student','tutor','admin') NOT NULL DEFAULT 'student'
- display_name VARCHAR(150)
- phone VARCHAR(32) NULL
- created_at DATETIME DEFAULT CURRENT_TIMESTAMP
- updated_at DATETIME ON UPDATE CURRENT_TIMESTAMP
- verified BOOLEAN DEFAULT FALSE

Indexes: UNIQUE(email), INDEX(role)

---

profiles

- id BIGINT PK
- user_id BIGINT FK -> users(id) UNIQUE
- bio TEXT
- location VARCHAR(255)
- language_native VARCHAR(50)
- language_learning VARCHAR(50)
- avatar_url VARCHAR(512)
- rating_avg DECIMAL(3,2) DEFAULT 0
- rating_count INT DEFAULT 0

Indexes: FK user_id

---

tutors

- id BIGINT PK
- user_id BIGINT FK -> users(id) UNIQUE
- approved BOOLEAN DEFAULT FALSE
- qualifications TEXT
- specialties TEXT
- created_at DATETIME

Notes: some systems store tutor info in profile; separate table allows approvals and verification flows.

---

courses

- id BIGINT PK
- tutor_id BIGINT FK -> tutors(id)
- title VARCHAR(255)
- description TEXT
- price_per_session DECIMAL(10,2)
- currency VARCHAR(8) DEFAULT 'KES'
- published BOOLEAN DEFAULT FALSE
- created_at DATETIME

Indexes: INDEX(tutor_id), INDEX(published)

---

sessions (bookings)

- id BIGINT PK
- course_id BIGINT FK -> courses(id) NULLABLE (null if ad-hoc)
- tutor_id BIGINT FK -> tutors(id)
- student_id BIGINT FK -> users(id)
- start_time DATETIME
- end_time DATETIME
- status ENUM('scheduled','cancelled','completed','no_show') DEFAULT 'scheduled'
- paid BOOLEAN DEFAULT FALSE
- booked_at DATETIME DEFAULT CURRENT_TIMESTAMP

Indexes: INDEX(tutor_id,start_time), INDEX(student_id,start_time), INDEX(status)

---

payments

- id BIGINT PK
- session_id BIGINT FK -> sessions(id)
- payer_id BIGINT FK -> users(id)
- amount DECIMAL(10,2)
- currency VARCHAR(8)

- provider VARCHAR(50) -- e.g., M-Pesa, Stripe
- provider_txn_id VARCHAR(255)
- status ENUM('pending','completed','failed','refunded')
- created_at DATETIME

Indexes: INDEX(session_id), INDEX(payer_id), UNIQUE(provider, provider_txn_id)

---

messages

- id BIGINT PK
- from_user_id BIGINT FK -> users(id)
- to_user_id BIGINT FK -> users(id)
- session_id BIGINT FK -> sessions(id) NULL
- body TEXT
- sent_at DATETIME DEFAULT CURRENT_TIMESTAMP
- read BOOLEAN DEFAULT FALSE

Indexes: INDEX(from_user_id,to_user_id), INDEX(session_id)

---

reviews

- id BIGINT PK
- session_id BIGINT FK -> sessions(id) NULL
- reviewer_id BIGINT FK -> users(id)
- reviewee_id BIGINT FK -> users(id)
- rating TINYINT NOT NULL CHECK (rating BETWEEN 1 AND 5)
- comment TEXT
- created_at DATETIME

Indexes: INDEX(reviewee_id), INDEX(reviewer_id)

---

resources

- id BIGINT PK
- course_id BIGINT FK -> courses(id) NULL
- uploaded_by BIGINT FK -> users(id)
- title VARCHAR(255)
- url VARCHAR(1024)
- type VARCHAR(50) -- 'pdf','video','link' etc
- uploaded_at DATETIME

Indexes: INDEX(course_id)

---

availabilities

- id BIGINT PK
- tutor_id BIGINT FK -> tutors(id)
- weekday TINYINT (0=Sun..6=Sat) or DATE for specific day
- start_time TIME
- end_time TIME
- timezone VARCHAR(64)

Notes: for recurring weekly availability, use weekday; for ad-hoc, store specific date instead.

---

notifications

- id BIGINT PK
- user_id BIGINT FK
- type VARCHAR(100)
- payload JSON
- read BOOLEAN DEFAULT FALSE
- created_at DATETIME

Indexes: INDEX(user_id, read)

---

4) UML class diagram (Mermaid)

```
classDiagram
    class User {
        +Long id
        +String email
        +String passwordHash
        +Role role
        +String displayName
        +DateTime createdAt
    }

    class Profile {
        +Long id
        +Long userId
        +String bio
        +String avatarUrl
        +Decimal ratingAvg
    }

    class Tutor {
        +Long id
        +Long userId
        +Boolean approved
        +String qualifications
    }

    class Course {
        +Long id
        +Long tutorId
        +String title
        +Decimal pricePerSession
    }

    class Session {
        +Long id
        +Long courseId
        +Long tutorId
        +Long studentId
        +DateTime startTime
        +DateTime endTime
        +SessionStatus status
    }

    class Payment {
        +Long id
        +Long sessionId
        +Long payerId
        +Decimal amount
        +String provider
    }

    class Message {
        +Long id
        +Long fromUserId
        +Long toUserId
        +String body
        +DateTime sentAt
    }

    User "1" -- "1" Profile : has
    User "1" -- "0..1" Tutor : mayBe
    Tutor "1" -- "*" Course : creates
    Course "1" -- "*" Session : schedules
    Session "*" -- "0..1" Payment : paidBy
    User "1" -- "*" Message : sends
```

5) Sequence diagram: Book a session (student books a tutor session)

```
sequenceDiagram
    participant Student
    participant Frontend
    participant Backend
    participant PaymentGateway
    participant Tutor

    Student->>Frontend: open course page & select slot
    Frontend->>Backend: request available slot & create session draft
    Backend-->>Frontend: draft session (status: pending)
    Frontend->>PaymentGateway: charge student
    PaymentGateway-->>Frontend: payment success
    Frontend->>Backend: confirm session + mark paid
    Backend-->>Student: confirmation + notification
    Backend-->>Tutor: notify new booking
```

---

6) Additional implementation notes

- Use UUIDs for public-facing references (e.g., booking codes) while keeping integer PKs internally for joins and performance.
- Payments: store provider and provider_txn_id for reconciliation. Keep PCI compliance in mind — do not store card data.
- For chat/messages scale: consider a separate message store (e.g., Redis, Kafka + Cassandra) if expecting heavy realtime traffic.
- Search & discovery: index courses by title, tutor specialties and language. Use a dedicated search engine (ElasticSearch) for advanced filters.
- Use foreign key constraints and ON DELETE CASCADE where appropriate (e.g., delete resources when a course is deleted) but be careful with user deletion — prefer soft deletes.

---

7) Next steps / options I can produce now

- Generate SQL DDL for MySQL (CREATE TABLE statements).
- Export diagrams as PNG/SVG.
- Produce a simplified ERD image (PNG) or PlantUML text.
- Tailor schema for multi-tenancy or scaling for messaging/video sessions.

Tell me which artifact you want next and I will generate it (SQL DDL, PNG export, or PlantUML).