Technology Stack for the Skill Swap App

Here’s a **recommended technology stack** for your app:

1. Backend (Server & API)

- **Node.js + Express.js** → For handling API requests.
- **MongoDB (Mongoose ORM)** → NoSQL database to store users, skills, and matches.
- **Firebase Authentication / JSON Web Tokens (JWT)** → For user authentication.
- **Socket.io** → For real-time messaging between users.

2. Frontend (Mobile App)

- **React Native (Expo or CLI)** → For cross-platform mobile development (iOS & Android).
- **Redux Toolkit or React Query** → For state management.
- **Axios** → For API requests.

3. Hosting & Deployment

- **Backend**: **Render/Vercel/AWS EC2** (for Node.js server)
- **Database**: **MongoDB Atlas** (cloud-based MongoDB)
- **Mobile App**: **Google Play Store & Apple App Store**
- **Real-time Chat**: **Firebase Firestore or Socket.io**

Entities, Attributes & Relationships

Below is a breakdown of the **main entities**, their **attributes**, and their **relationships**.

1. User Entity (Users who want to learn or teach)

- `user_id` (Unique identifier)
- `name`
- `email`
- `password`
- `phone_number`
- `location` (City, Country)
- `bio`
- `profile_picture`
- `skills_offered` (Array of skills they can teach)
- `skills_wanted` (Array of skills they want to learn)
- `rating` (Average rating from past exchanges)
- `availability` (Days/Time available)

🔹 **Relationships**:

- A **User** can teach multiple **Skills**.
- A **User** can learn multiple **Skills**.

- A **User** can have many **Reviews**.

---

2. Skill Entity (Skills available for swapping)

- `skill_id`
- `name` (e.g., "Graphic Design", "French", "Public Speaking")
- `category` (e.g., Tech, Language, Business)
- `description`
- `popularity` (Number of users offering/learning)

🔹 **Relationships**:

- A **Skill** can be offered by multiple **Users**.
- A **Skill** can be requested by multiple **Users**.

---

3. Skill Swap Match Entity (Pairing users for learning & teaching)

- `match_id`
- `teacher_id` (User who will teach)
- `learner_id` (User who will learn)
- `skill_id` (The skill being exchanged)
- `status` (Pending, Accepted, Completed)
- `scheduled_time` (When they will meet)
- `mode` (In-person or Online)

🔹 **Relationships**:

- A **Match** links **two Users** through a **Skill**.

---

4. Reviews & Ratings Entity (User feedback after skill swap)

- `review_id`
- `reviewer_id` (User giving the review)
- `reviewed_id` (User receiving the review)
- `rating` (1-5 stars)
- `comment`
- `date`

🔹 **Relationships**:

- A **User** can give multiple **Reviews**.
- A **User** can receive multiple **Reviews**.

---

5. Messages Entity (Real-time chat for communication)

- `message_id`
- `sender_id`
- `receiver_id`
- `message_content`
- `timestamp`
- `read_status`

⬍ **Relationships**:

- A **User** can send messages to another **User**.

How These Entities Work Together

1. A **User** signs up and adds the **Skills** they can teach & the ones they want to learn.
2. The system finds **Matches** based on skills.
3. Once a match is confirmed, users schedule a session (either in-person or online).
4. After the session, users leave **Reviews & Ratings**.
5. Users can chat via **Messages**.