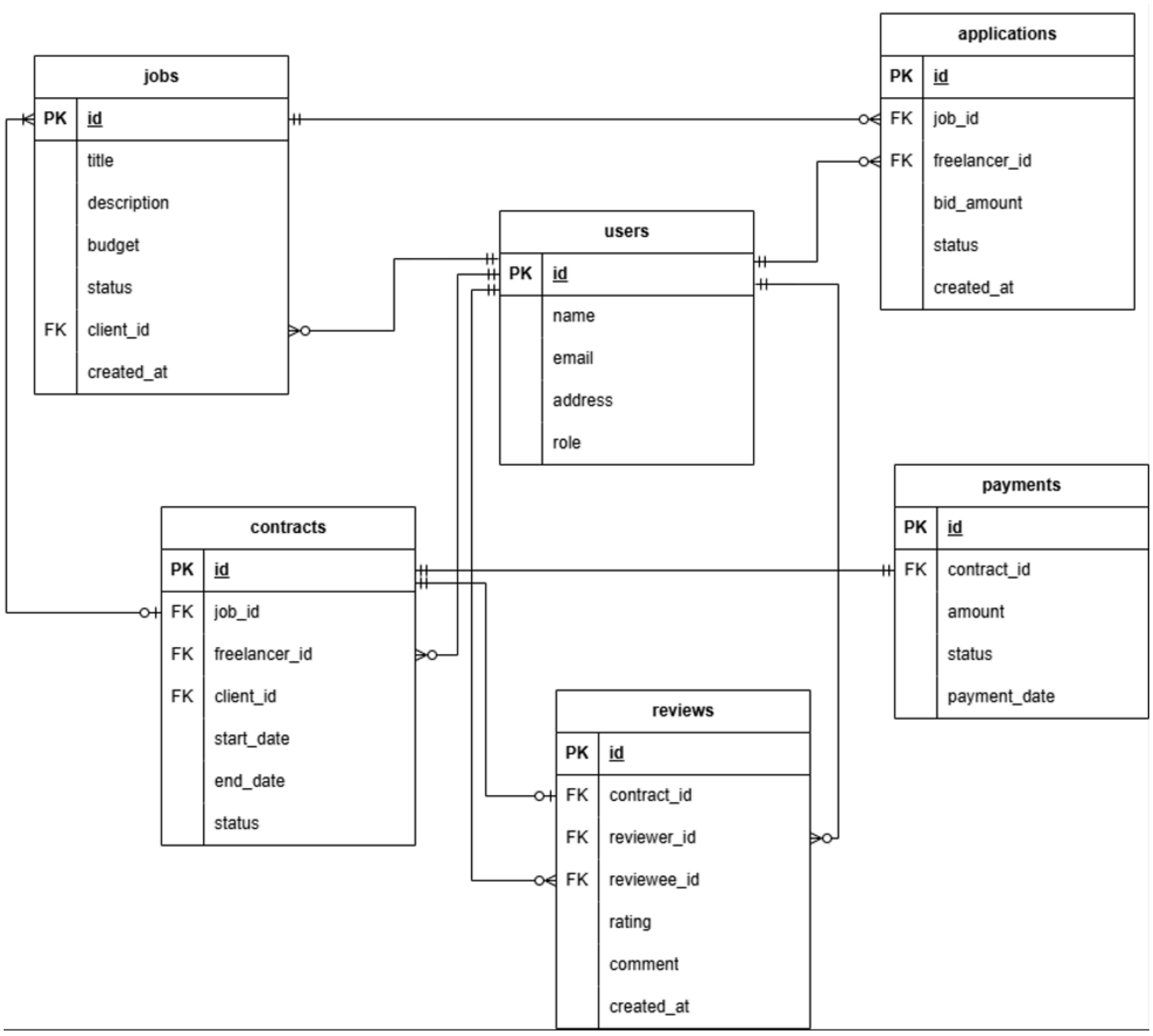
Entity Relationship Diagram and Explanation



Chosen Database System: PostgreSQL

For this project, I have chosen **PostgreSQL**, an advanced open-source **relational database management system (RDBMS)** known for its strong reliability, extensibility, and standards compliance. PostgreSQL is an excellent choice for a **Freelancer Job Board API** because it provides robust support for complex queries, data integrity, and scalability.

Why PostgreSQL?

1. **Relational Structure & ACID Compliance:**
   PostgreSQL is an **ACID-compliant** (Atomicity, Consistency, Isolation, Durability) database, ensuring that transactions are executed reliably. This is crucial for a job board where data integrity is essential—for example, ensuring that jobs, applications, and payments are correctly processed without data corruption.

2. **SQL Compliance & Extensibility:**
   PostgreSQL follows the **SQL standard** while offering powerful extensions like **PostGIS (for geolocation)** and **JSONB** (for semi-structured data). It allows a mix of structured and unstructured data, making it flexible.

3. **Scalability & Performance:**
   - PostgreSQL handles large-scale applications efficiently with indexing, partitioning, and query optimisation.
   - It supports **horizontal and vertical scaling**, making it ideal for a growing application with increasing freelancer-client interactions.
4. **Strong Security Features:**
   PostgreSQL provides **role-based access control (RBAC)**, SSL encryption, and built-in **authentication methods** (e.g., password-based, certificate-based, or external authentication).

5. **ORM Compatibility (SQLAlchemy):**
   Since I am using **SQLAlchemy**, PostgreSQL integrates seamlessly with it, allowing me to interact with the database in an efficient and structured way.

Comparison with Other Database Systems

To justify the choice of PostgreSQL, let's compare it with other popular database types:

1. PostgreSQL vs MySQL (Relational DB)

Feature	PostgreSQL	MySQL
ACID Compliance	Fully ACID-compliant	ACID-compliant but with some exceptions
Complex Queries	Handles complex queries well	Optimized for read-heavy workloads
JSON Support	JSON & JSONB for semi-structured data	Basic JSON support (less efficient)
Extensibility	Supports procedural languages, full-text search, and indexing	Limited extensibility
Concurrency Control	Uses MVCC (better concurrency & performance)	Uses table-level locking more often

Verdict: PostgreSQL is superior for complex queries, extensibility, and JSON data handling. MySQL is simpler but less feature-rich.

---

2. PostgreSQL vs MongoDB (NoSQL)

Feature	PostgreSQL	MongoDB
Data Structure	Relational (tables & rows)	Document-based (JSON-like BSON)
Schema Enforcement	Strongly enforced schema (structured data)	Schema-less (flexible data)
Query Language	SQL	MongoDB Query Language (MQL)
Transactions	ACID transactions	Multi-document transactions (added later)
Scalability	Vertical & horizontal scaling	High horizontal scaling

Verdict: If my project required highly dynamic or semi-structured data (e.g., social media feeds), MongoDB would be useful. However, since I need **structured relationships (Users, Jobs, Applications, Payments)** with strong data integrity, PostgreSQL is the better choice.

---

3. PostgreSQL vs Firebase (Realtime NoSQL)

Feature	PostgreSQL	Firebase
Type	Relational SQL Database	NoSQL Realtime Database
Schema	Structured & strict	Unstructured & flexible
Query Power	Advanced SQL & indexing	Basic querying
Performance	Optimized for structured data	Optimized for real-time syncing
Offline Mode	Not natively designed for offline mode	Designed for mobile apps with offline syncing

Verdict: Firebase is ideal for **real-time applications like messaging** but not suitable for a structured job board that requires **complex queries, relationships, and transactions**.

Final Justification for PostgreSQL

Considering the project requirements—structured job listings, user accounts, applications, and financial transactions—**PostgreSQL is the best fit** because:

- It ensures **data integrity** and **supports relationships** between entities.
- It allows **complex queries** for filtering jobs, applications, and transactions.
- It scales well with indexing and optimization features.
- It integrates well with **SQLAlchemy**, making ORM-based database management easier.