



# CollabSpace Joint PostgreSQL Schema (Improved)

*This unified schema shows all data relationships between services. In reality, each backend owns its section — they just share `user_id` and `project_id` references for integration.*

---

```
-- =====
-- ■ AUTH & USER MANAGEMENT (Node Gateway)
-- =====

CREATE TYPE user_role AS ENUM ('admin', 'member');

CREATE TABLE users (
    id                UUID PRIMARY KEY DEFAULT gen_random_uuid()
    full_name         VARCHAR(100) NOT NULL,
    email             VARCHAR(100) UNIQUE NOT NULL,
    password_hash     TEXT NOT NULL,
    avatar_url        TEXT,
    role              user_role DEFAULT 'member',
    is_active         BOOLEAN DEFAULT TRUE,
    created_at        TIMESTAMP DEFAULT NOW(),
    updated_at        TIMESTAMP DEFAULT NOW()
);

CREATE TABLE refresh_tokens (
    id                UUID PRIMARY KEY DEFAULT gen_random_uuid()
    user_id           UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE
    token             TEXT UNIQUE NOT NULL,
    expires_at        TIMESTAMP NOT NULL,
    created_at        TIMESTAMP DEFAULT NOW()
);

-- =====
-- ■ PROJECTS, TASKS & TEAMS (Java Service)
```

-- =====

```
CREATE TYPE project_status AS ENUM ('active', 'completed', 'archived');
CREATE TYPE member_role AS ENUM ('owner', 'admin', 'member');
CREATE TYPE task_status AS ENUM ('todo', 'in_progress', 'done');
CREATE TYPE task_priority AS ENUM ('low', 'medium', 'high');
```

```
CREATE TABLE projects (
    id                UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    name              VARCHAR(150) NOT NULL,
    description        TEXT,
    created_by        UUID NOT NULL REFERENCES users(id),
    status             project_status DEFAULT 'active',
    start_date         DATE NOT NULL,
    end_date           DATE,
    created_at         TIMESTAMP DEFAULT NOW()
);
```

```
CREATE TABLE project_members (
    id                UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    project_id         UUID NOT NULL REFERENCES projects(id) ON DELETE CASCADE,
    user_id            UUID NOT NULL REFERENCES users(id),
    role               member_role DEFAULT 'member',
    joined_at          TIMESTAMP DEFAULT NOW()
);
```

```
CREATE TABLE tasks (
    id                UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    project_id         UUID NOT NULL REFERENCES projects(id) ON DELETE CASCADE,
    title              VARCHAR(150) NOT NULL,
    description        TEXT,
    assignee_id        UUID REFERENCES users(id),
    status             task_status DEFAULT 'todo',
    priority           task_priority DEFAULT 'medium',
    due_date           DATE,
    created_at         TIMESTAMP DEFAULT NOW(),
    updated_at         TIMESTAMP DEFAULT NOW()
);
```

```
CREATE TABLE teams (
    id                UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    name              VARCHAR(100),
    members            UUID[] NOT NULL,
```

```

        project_ids      UUID[] DEFAULT '{}',
        last_refreshed   TIMESTAMP DEFAULT NOW()
    );

-- =====
-- ■ COMMENTS, CHAT & NOTIFICATIONS (Node Realtime)
-- =====

CREATE TYPE chat_type AS ENUM ('direct', 'group', 'project');
CREATE TYPE chat_role AS ENUM ('member', 'admin');
CREATE TYPE notification_type AS ENUM ('task_assigned', 'ment:

CREATE TABLE chats (
    id                UUID PRIMARY KEY DEFAULT gen_random_uuid(
    type              chat_type DEFAULT 'project',
    name              VARCHAR(100),
    created_by        UUID NOT NULL REFERENCES users(id),
    created_at        TIMESTAMP DEFAULT NOW()
);

CREATE TABLE chat_members (
    chat_id           UUID NOT NULL REFERENCES chats(id) ON DEL
    user_id           UUID NOT NULL REFERENCES users(id),
    role              chat_role DEFAULT 'member',
    joined_at         TIMESTAMP DEFAULT NOW(),
    PRIMARY KEY (chat_id, user_id)
);

CREATE TABLE messages (
    id                UUID PRIMARY KEY DEFAULT gen_random_uuid(
    chat_id           UUID NOT NULL REFERENCES chats(id) ON DEL
    sender_id         UUID NOT NULL REFERENCES users(id),
    content            TEXT NOT NULL,
    attachment_url    TEXT,
    created_at        TIMESTAMP DEFAULT NOW(),
    edited_at         TIMESTAMP
);

CREATE TABLE comments (
    id                UUID PRIMARY KEY DEFAULT gen_random_uuid(
    project_id        UUID NOT NULL REFERENCES projects(id) ON
    author_id         UUID NOT NULL REFERENCES users(id),
    content            TEXT NOT NULL,

```

```

        created_at      TIMESTAMP DEFAULT NOW(),
        edited_at       TIMESTAMP
    );




CREATE TABLE notifications (
    id                  UUID PRIMARY KEY DEFAULT gen_random_uuid()
    user_id            UUID NOT NULL REFERENCES users(id),
    type               notification_type NOT NULL,
    message            TEXT NOT NULL,
    is_read            BOOLEAN DEFAULT FALSE,
    created_at         TIMESTAMP DEFAULT NOW()
);

```

---



## Division of Responsibility

Service	Database Ownership	Description
 <b>Node Gateway (Auth)</b>	<code>users,</code> <code>refresh_tokens</code>	Handles all authentication, JWT issuing, and user identity. This is the <b>entry gate</b> to every other service.
 <b>Java Service (Projects)</b>	<code>projects,</code> <code>project_members,</code> <code>tasks, teams</code>	Core business domain. Manages projects, their members, and tasks. Also maintains lightweight “teams” as cached groupings.
 <b>Node Realtime (Social Layer)</b>	<code>chats, chat_members,</code> <code>messages, comments,</code> <code>notifications</code>	Real-time engine. Responsible for communication (chat), project comments, and in-app notifications. Uses Socket.IO for live updates.

---



## Cross-Service Integration Rules

Relationship	Origin	Consumer	Description
<code>users.id</code>	Gateway	Java + Realtime	Universal identifier for authentication. All other services trust it through JWT.
<code>projects.id</code>	Java	Realtime	Used for project-linked chats/comments.
<code>assignee_id</code>	Java	Gateway	Mapped to users for project task assignment.
<code>sender_id</code>	Realtime	Gateway	Refers to the message sender identity.



## Quick Summary of the Final MVP Stack

Layer	Focus	Key Entities	Stack	Scaling
<b>Node Gateway</b>	Auth & Access	Users, Tokens	Node + Prisma	Stateless scaling
<b>Java Service</b>	Project Logic	Projects, Tasks, Teams	Spring Boot + JPA	Vertical scaling
<b>Node Realtime</b>	Collaboration Layer	Chats, Comments, Notifications	Node + Socket.IO + Redis	Horizontal scaling