GazozHub Database Technical Documentation

This documentation provides a comprehensive overview of the **GazozHub** database architecture, designed to support real-time academic collaboration, project management, and asset tracking. The system utilizes **PostgreSQL** to maintain strict relational integrity and data persistence.

---

1. Database Overview & Connection

The database is managed using the pg (PostgreSQL) library in a Node.js environment. It utilizes a **Connection Pool** to handle multiple concurrent requests efficiently.

- **Host:** Configured via environment variables (PGHOST) or connection strings.
- **SSL Support:** Enabled for production environments like Heroku or Render to ensure secure data transit.
- **Initialization:** The system includes an ensureTables function to automatically bootstrap the schema if tables do not exist.

---

2. Entity-Relationship Schema

The database structure is built around several core tables that manage users, projects, tasks, and collaborative content.

2.1 Table: users

This table stores primary identity and authentication data for all platform participants. | Column | Data Type | Constraints | Description | | :--- | :--- | :--- | :--- | | id | SERIAL | PRIMARY KEY | Unique identifier for each user. | | username | TEXT | UNIQUE, NOT NULL | Display name used for login and identification. | | email | TEXT | UNIQUE, NOT NULL | Primary contact and login email. | | password_hash | TEXT | NOT NULL | Bcrypt-hashed password for secure storage. | | role | TEXT | DEFAULT 'student' | System access level (e.g., 'admin', 'student'). | | created_at | TIMESTAMP | DEFAULT NOW() | Timestamp of account creation. |

2.2 Table: projects

Stores metadata for digital workspaces created by users. | Column | Data Type | Constraints | Description | | :--- | :--- | :--- | :--- | | id | SERIAL | PRIMARY KEY | Unique identifier for the project. | | owner_id | INTEGER | REFERENCES users(id) | The primary creator and owner of the project. | | name | TEXT | NOT NULL | The display name of the project. | | description | TEXT | NULLABLE | Brief summary of project goals. | | is_public | BOOLEAN | DEFAULT FALSE | Controls if the project is visible to non-members. | | is_tasks_public | BOOLEAN | DEFAULT FALSE | Controls public visibility specifically for the Kanban board. |

2.3 Table: project_members

A junction table facilitating the many-to-many relationship between users and projects. | Column | Data Type | Constraints | Description | | :--- | :--- | :--- | :--- | | project_id | INTEGER | FK (projects.id) | The associated project. | | user_id | INTEGER | FK (users.id) | The associated user member. | | role | TEXT | NOT NULL | Member-specific role: 'owner', 'editor', or 'viewer'. | | joined_at | TIMESTAMP | DEFAULT NOW() | When the user joined the project team. |

2.4 Table: project_tasks

Stores Kanban board items and their current workflow status. | Column | Data Type | Constraints | Description | | :--- | :--- | :--- | :--- | | id | SERIAL | PRIMARY KEY | Unique task identifier. | | project_id | INTEGER | FK (projects.id) | The project containing this task. | | title | TEXT | NOT NULL | Name/Title of the task. | | status | TEXT | DEFAULT 'todo' | Status: 'todo', 'in_progress', or 'done'. | | assignee_id | INTEGER | REFERENCES users(id) | The user responsible for the task. |

---

3. Storage & Integrity Logic

3.1 Relational Integrity

- **Cascading Deletes:** The database uses ON DELETE CASCADE on project foreign keys. If a project is deleted, all its tasks, files, and member records are automatically purged to prevent orphaned data.
- **Transactions:** Critical operations like project creation use SQL Transactions (BEGIN, COMMIT, ROLLBACK) to ensure that the project and its initial owner role are created atomically.

3.2 File & Metadata Management

GazozHub uses a hybrid storage approach for project assets:

- **Physical Storage:** Files are stored on the server disk within the uploads/ directory using unique timestamps to prevent filename collisions.
- **Metadata:** The project_files table stores the file_path, filename, file_type, and the uploader_id for retrieval and auditing.

---

4. Query Examples (Business Logic)

- **Fetching My Projects:** Retrieves projects where the current user is the owner.
- **Shared Project Access:** Retrieves projects where the user is a member but not the owner, or projects marked as public.
- **Admin Dashboard Stats:** Aggregates user roles and registration counts to provide a system-wide overview.