Database Schema Documentation

xAI

May 5, 2025

Contents

1 Introduction

This document provides detailed documentation for the database schema comprising four tables: `users`, `projects`, `bugs`, and `messages`. The schema is designed for a project management and bug tracking system, supporting user management, project tracking, bug reporting, and messaging functionalities. The database is implemented in MySQL with the InnoDB storage engine to support foreign key relationships.

2 Database Overview

The schema includes the following tables:

- `users`: Stores information about registered users.
- `projects`: Manages project details, including titles, types, and client information.
- `bugs`: Tracks bugs associated with projects, including their status and assignment.
- `messages`: Stores messages sent by users, linked to their user accounts.

The tables are interrelated through foreign keys to ensure data integrity and support efficient querying.

3 Table Details

3.1 Users Table

The `users` table stores information about registered users, including their credentials and roles.

Column	Data Type	Constraints	Description
`user_id`	`INT`	`AUTO_INCREMENT PRIMARY KEY NOT NULL`	Unique identifier for each user.
`user_fullname`	`VARCHAR(256)`	`NULL`	Full name of the user (optional).
`user_email`	`VARCHAR(256)`	`NOT NULL UNIQUE`	Unique email address for user authentication.
`user_pwd`	`VARCHAR(256)`	`NOT NULL`	Hashed password for user authentication.
`role`	`ENUM('customer', 'admin', 'developer')`	`NOT NULL DEFAULT 'customer'`	User role defining access level.
`created_at`	`DATETIME`	`DEFAULT CURRENT_TIMESTAMP`	Timestamp of user record creation.

Notes:

- The `user_email` is unique to prevent duplicate accounts.

- Passwords in `user_pwd` should be hashed (e.g., using bcrypt) before storage.
- The `role` field uses `ENUM` to restrict values to predefined roles.

3.2 Projects Table

The `projects` table stores details about projects, including their titles, types, and associated clients.

Column	Data Type	Constraints	Description
`project_id`	`INT`	`AUTO_INCREMENT PRIMARY KEY NOT NULL`	Unique identifier for each project.
`project_title`	`VARCHAR(300)`	`NOT NULL`	Title of the project.
`project_type`	`ENUM('web', 'mobile', 'desktop')`	`NOT NULL`	Type of project.
`project_description`	`TEXT`	`NOT NULL`	Detailed description of the project.
`client_name`	`VARCHAR(200)`	`NOT NULL`	Name of the client associated with the project.
`created_at`	`DATETIME`	`DEFAULT CURRENT_TIMESTAMP`	Timestamp of project record creation.

Notes:

- The `project_type` uses `ENUM` to restrict values to 'web', 'mobile', or 'desktop'.
- The `project_description` uses `TEXT` to accommodate lengthy descriptions.

3.3 Bugs Table

The `bugs` table tracks issues reported for projects, including their status, priority, and assignment.

Column	Data Type	Constraints	Description
`id`	`INT`	`AUTO_INCREMENT PRIMARY KEY NOT NULL`	Unique identifier for each bug.
`bug_name`	`VARCHAR(120)`	`NOT NULL`	Name or title of the bug.
`project_id`	`INT`	`NOT NULL FOREIGN KEY`	References `project_id` in `projects` table.
`category`	`TEXT`	`NOT NULL`	Category of the bug (e.g., 'UI', 'backend').

details	TEXT	NULL	Detailed description of the bug (optional).
assigned_to	INT	NULL FOREIGN KEY	References user_id in users table (optional).
status	ENUM('open', 'in_progress', 'closed')	DEFAULT 'open'	Current status of the bug.
priority	ENUM('low', 'medium', 'high')	DEFAULT 'medium'	Priority level of the bug.
created_at	DATETIME	DEFAULT CURRENT_TIMESTAMP	Timestamp of bug record creation.

Indexes:

- INDEX idx_project_id (project_id): Improves query performance for project-related searches.

Notes:

- The project_id and assigned_to fields enforce referential integrity via foreign keys.
- The status and priority fields use ENUM to ensure consistent values.

3.4 Messages Table

The messages table stores user-submitted messages, linked to their user accounts.

Column	Data Type	Constraints	Description
id	INT	AUTO_INCREMENT PRIMARY KEY NOT NULL	Unique identifier for each message.
user_id	INT	NOT NULL FOREIGN KEY	References user_id in users table.
email	VARCHAR(300)	NOT NULL	Email address of the message sender.
message	TEXT	NOT NULL	Content of the message.
status	ENUM('unread', 'read', 'archived')	DEFAULT 'unread'	Status of the message.
created_at	DATETIME	DEFAULT CURRENT_TIMESTAMP	Timestamp of message record creation.

Indexes:

- INDEX idx_user_id (user_id): Improves query performance for user-related searches.

Notes:

- The `user_id` field links messages to registered users, ensuring traceability.
- The `email` field may be redundant if it matches `user_email` in the `users` table.

4 Relationships

The tables are interconnected through foreign keys to maintain data integrity:

- `bugs.project_id` references `projects.project_id`: Links bugs to their respective projects.
- `bugs.assigned_to` references `users.user_id`: Links bugs to assigned users (optional).
- `messages.user_id` references `users.user_id`: Links messages to their senders.

5 Usage Notes

- **Execution Order**: Create tables in the order `users`, `projects`, `bugs`, `messages` due to foreign key dependencies.
- **Storage Engine**: Use InnoDB to support foreign keys (`ENGINE=InnoDB`).
- **Security**: Hash passwords in `users.user_pwd` using a secure algorithm (e.g., bcrypt).
- **Data Validation**: Validate `user_email` format and ensure `project_type`, `role`, `status`, and `priority` adhere to `ENUM` values in the application layer.
- **Performance**: Indexes on `bugs.project_id` and `messages.user_id` optimize queries involving joins.

6 Sample SQL Schema

The following SQL script creates the complete schema:

```
CREATE TABLE users (
    user_id INT AUTO_INCREMENT PRIMARY KEY NOT NULL,
    user_fullname VARCHAR(256),
    user_email VARCHAR(256) NOT NULL UNIQUE,
    user_pwd VARCHAR(256) NOT NULL,
    role ENUM('customer', 'admin', 'developer') NOT NULL DEFAULT 'customer',
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP
) ENGINE=InnoDB;

CREATE TABLE projects (
    project_id INT AUTO_INCREMENT PRIMARY KEY NOT NULL,
    project_title VARCHAR(300) NOT NULL,
```

```
    project_type ENUM('web', 'mobile', 'desktop') NOT NULL,
    project_description TEXT NOT NULL,
    client_name VARCHAR(200) NOT NULL,
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP
) ENGINE=InnoDB;

CREATE TABLE bugs (
    id INT AUTO_INCREMENT PRIMARY KEY NOT NULL,
    bug_name VARCHAR(120) NOT NULL,
    project_id INT NOT NULL,
    category TEXT NOT NULL,
    details TEXT,
    assigned_to INT,
    status ENUM('open', 'in_progress', 'closed') DEFAULT 'open',
    priority ENUM('low', 'medium', 'high') DEFAULT 'medium',
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (project_id) REFERENCES projects(project_id),
    FOREIGN KEY (assigned_to) REFERENCES users(user_id),
    INDEX idx_project_id (project_id)
) ENGINE=InnoDB;

CREATE TABLE messages (
    id INT AUTO_INCREMENT PRIMARY KEY NOT NULL,
    user_id INT NOT NULL,
    email VARCHAR(300) NOT NULL,
    message TEXT NOT NULL,
    status ENUM('unread', 'read', 'archived') DEFAULT 'unread',
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES users(user_id),
    INDEX idx_user_id (user_id)
) ENGINE=InnoDB;
```

7 Conclusion

This schema provides a robust foundation for a project management and bug tracking system. It ensures data integrity through foreign keys, optimizes performance with indexes, and uses `ENUM` for consistent data entry. Future enhancements may include adding `updated_at` timestamps, soft delete functionality, or additional audit fields.