ROUND-1 ASSESSMENT

Intern Assignment: Build a Project Management Tool

Candidate Information

Name:	G Mahesh
Program:	MCA
Roll Number:	202468075
Email:	g.mahesh7735@gmail.com

Assignment Overview

Objective: Build a simplified Project Management Tool that allows users to manage projects, tasks, teams, and track progress.

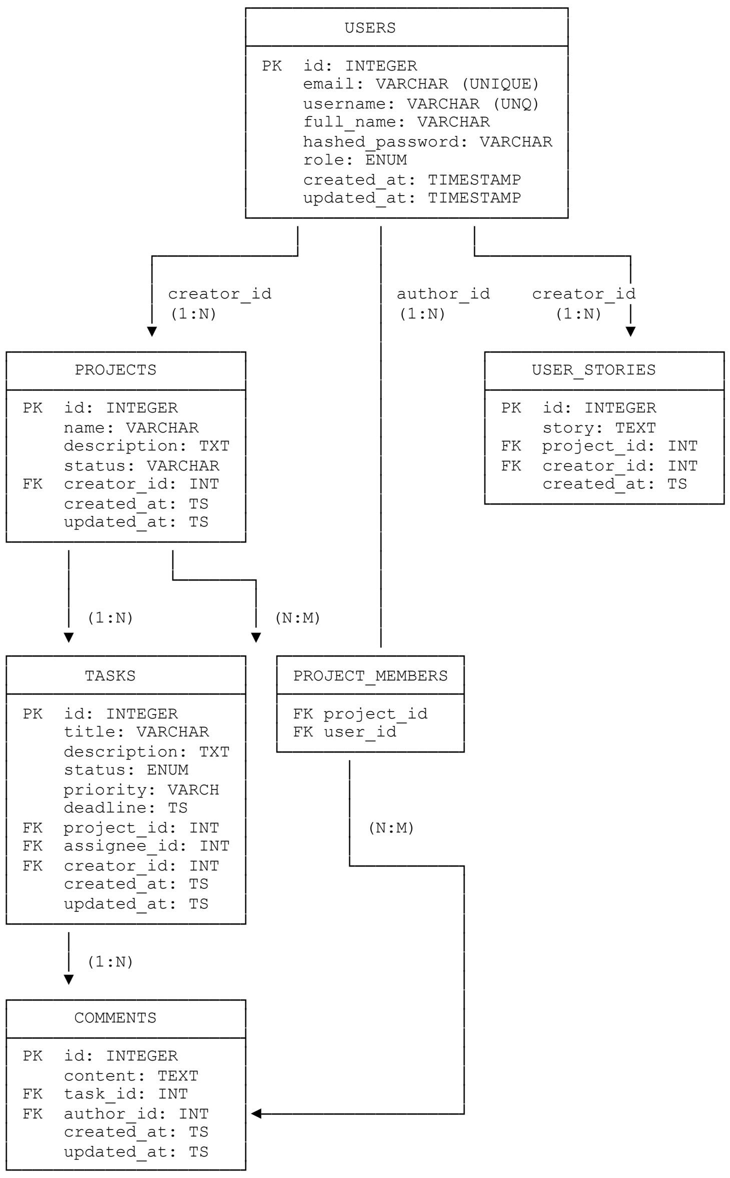
Key Features Implemented:

- Project creation and assignment
- Task management (create, update, delete)
- User management (Admin, Manager, Developer roles)
- Task status tracking (To Do, In Progress, Done)
- Deadline handling and metrics
- **Bonus:** AI-Powered User Story Generator using GROQ

Tech Stack:

- **Backend:** Python with Flask
- **Database:** PostgreSQL
- **Frontend:** React
- **Authentication:** JWT token-based security

1. Visual ER Diagram



```
USERS
PK  id: INTEGER
    email: VARCHAR (UNIQUE)
    username: VARCHAR (UNQ)
    full_name: VARCHAR
    hashed_password: VARCHAR
    role: ENUM
    created_at: TIMESTAMP
    updated_at: TIMESTAMP

PROJECTS
PK  id: INTEGER
    name: VARCHAR
    description: TXT
    status: VARCHAR
FK  creator_id: INT
    created_at: TS
    updated_at: TS

USER_STORIES
PK  id: INTEGER
    story: TEXT
FK  project_id: INT
FK  creator_id: INT
    created_at: TS

TASKS
PK  id: INTEGER
    title: VARCHAR
    description: TXT
    status: ENUM
    priority: VARCH
    deadline: TS
FK  project_id: INT
FK  assignee_id: INT
FK  creator_id: INT
    created_at: TS
    updated_at: TS

PROJECT_MEMBERS
FK project_id
FK user_id

COMMENTS
PK  id: INTEGER
    content: TEXT
FK  task_id: INT
FK  author_id: INT
    created_at: TS
    updated_at: TS
```

```
Legend: PK=Primary Key, FK=Foreign Key, TS=TIMESTAMP,
        TXT=TEXT, INT=INTEGER, VARCH=VARCHAR, UNQ=UNIQUE
```

2. Detailed Table Descriptions

2.1 USERS Table

Purpose: Stores all user information with role-based access control

Column	Type	Constraints	Description
id	INTEGER	PRIMARY KEY, AUTO_INCREMENT	Unique user identifier
email	VARCHAR(255)	UNIQUE, NOT NULL	User's email address
username	VARCHAR(100)	UNIQUE, NOT NULL	User's login username
full_name	VARCHAR(255)	NOT NULL	User's full name
hashed_password	VARCHAR(255)	NOT NULL	Bcrypt hashed password
role	ENUM	NOT NULL	User role: Admin, Manager, Developer
created_at	TIMESTAMP	DEFAULT NOW()	Account creation timestamp
updated_at	TIMESTAMP	ON UPDATE NOW()	Last update timestamp

Indexes:

- PRIMARY KEY on **id**
- UNIQUE INDEX on **email**
- UNIQUE INDEX on **username**

Relationships:

- One-to-Many with PROJECTS (as creator)
- One-to-Many with TASKS (as assignee)
- One-to-Many with TASKS (as creator)
- One-to-Many with COMMENTS (as author)
- One-to-Many with USER_STORIES (as creator)
- Many-to-Many with PROJECTS (as team member)

2.2 PROJECTS Table

Purpose: Stores project information and metadata

Column	**Type**	**Constraints**	**Description**
id	INTEGER	PRIMARY KEY, AUTO_INCREMENT	Unique project identifier
name	VARCHAR(255)	NOT NULL	Project name
description	TEXT	NULL	Project description
status	VARCHAR(50)	DEFAULT 'Active'	Project status
creator_id	INTEGER	FOREIGN KEY, ON DELETE SET NULL	User who created the project
created_at	TIMESTAMP	DEFAULT NOW()	Project creation timestamp
updated_at	TIMESTAMP	ON UPDATE NOW()	Last update timestamp

Indexes:

- PRIMARY KEY on **id**
- INDEX on **name**
- FOREIGN KEY on **creator_id** → USERS(id)

Relationships:

- Many-to-One with USERS (creator)
- One-to-Many with TASKS
- One-to-Many with USER_STORIES
- Many-to-Many with USERS (team members via PROJECT_MEMBERS)

2.3 TASKS Table

Purpose: Stores individual task information and assignments

Column	Type	Constraints	Description
id	INTEGER	PRIMARY KEY, AUTO_INCREMENT	Unique task identifier
title	VARCHAR(255)	NOT NULL	Task title
description	TEXT	NULL	Detailed task description
status	ENUM	NOT NULL, DEFAULT 'To Do'	Task status: To Do, In Progress, Done
priority	VARCHAR(50)	DEFAULT 'Medium'	Task priority: Low, Medium, High
deadline	TIMESTAMP	NULL	Task deadline
project_id	INTEGER	FOREIGN KEY, NOT NULL, ON DELETE CASCADE	Associated project
assignee_id	INTEGER	FOREIGN KEY, ON DELETE SET NULL	User assigned to task
creator_id	INTEGER	FOREIGN KEY, ON DELETE SET NULL	User who created task
created_at	TIMESTAMP	DEFAULT NOW()	Task creation timestamp
updated_at	TIMESTAMP	ON UPDATE NOW()	Last update timestamp

Indexes:

- PRIMARY KEY on **id**
- INDEX on **title**
- INDEX on **status**
- FOREIGN KEY on **project_id** → PROJECTS(id)
- FOREIGN KEY on **assignee_id** → USERS(id)
- FOREIGN KEY on **creator_id** → USERS(id)

Relationships:

- Many-to-One with PROJECTS
- Many-to-One with USERS (assignee)

- Many-to-One with USERS (creator)
- One-to-Many with COMMENTS

2.4 COMMENTS Table

Purpose: Stores comments/discussions on tasks

Column	Type	Constraints	Description
id	INTEGER	PRIMARY KEY, AUTO_INCREMENT	Unique comment identifier
content	TEXT	NOT NULL	Comment text content
task_id	INTEGER	FOREIGN KEY, NOT NULL, ON DELETE CASCADE	Associated task
author_id	INTEGER	FOREIGN KEY, NOT NULL, ON DELETE CASCADE	User who wrote comment
created_at	TIMESTAMP	DEFAULT NOW()	Comment creation timestamp
updated_at	TIMESTAMP	ON UPDATE NOW()	Last update timestamp

Indexes:

- PRIMARY KEY on **id**
- FOREIGN KEY on **task_id** → TASKS(id)
- FOREIGN KEY on **author_id** → USERS(id)

Relationships:

- Many-to-One with TASKS
- Many-to-One with USERS (author)

2.5 PROJECT_MEMBERS Table (Association Table)

Purpose: Many-to-Many relationship between PROJECTS and USERS (team members)

Column	Type	Constraints	Description
project_id	INTEGER	FOREIGN KEY, ON DELETE CASCADE	Project identifier
user_id	INTEGER	FOREIGN KEY, ON DELETE CASCADE	User identifier

Indexes:

- COMPOSITE PRIMARY KEY on (**project_id**, **user_id**)
- FOREIGN KEY on **project_id** → PROJECTS(id)
- FOREIGN KEY on **user_id** → USERS(id)

Relationships:

- Links PROJECTS and USERS in many-to-many relationship

2.6 USER_STORIES Table

Purpose: Stores AI-generated user stories for projects

Column	Type	Constraints	Description
id	INTEGER	PRIMARY KEY, AUTO_INCREMENT	Unique story identifier
story	TEXT	NOT NULL	User story text
project_id	INTEGER	FOREIGN KEY, NOT NULL, ON DELETE CASCADE	Associated project
creator_id	INTEGER	FOREIGN KEY, ON DELETE SET NULL	User who generated story
created_at	TIMESTAMP	DEFAULT NOW()	Story creation timestamp

Indexes:

- PRIMARY KEY on **id**
- FOREIGN KEY on **project_id** → PROJECTS(id)
- FOREIGN KEY on **creator_id** → USERS(id)

Relationships:

- Many-to-One with PROJECTS

- Many-to-One with USERS (creator)

3. Relationship Summary

3.1 One-to-Many Relationships

1. **USERS → PROJECTS** (as creator): One user can create multiple projects; each project has one creator
2. **PROJECTS → TASKS**: One project can have multiple tasks; each task belongs to one project
3. **USERS → TASKS** (as assignee): One user can be assigned multiple tasks; each task can be assigned to one user
4. **USERS → TASKS** (as creator): One user can create multiple tasks; each task has one creator
5. **TASKS → COMMENTS**: One task can have multiple comments; each comment belongs to one task
6. **USERS → COMMENTS** (as author): One user can write multiple comments; each comment has one author
7. **PROJECTS → USER_STORIES**: One project can have multiple user stories; each user story belongs to one project
8. **USERS → USER_STORIES** (as creator): One user can generate multiple user stories; each user story has one creator

3.2 Many-to-Many Relationships

1. **USERS ↔ PROJECTS** (team membership): One user can be a member of multiple projects; one project can have multiple team members. Implemented via PROJECT_MEMBERS association table

4. Database Constraints & Rules

4.1 Cascade Rules

ON DELETE CASCADE:

- When a PROJECT is deleted → all associated TASKS are deleted
- When a TASK is deleted → all associated COMMENTS are deleted
- When a PROJECT is deleted → all PROJECT_MEMBERS entries are deleted
- When a USER is deleted → all PROJECT_MEMBERS entries are deleted
- When a PROJECT is deleted → all USER_STORIES are deleted

ON DELETE SET NULL:

- When a USER (creator) is deleted → PROJECT.creator_id is set to NULL

- When a USER (assignee) is deleted → TASK.assignee_id is set to NULL
- When a USER (creator) is deleted → TASK.creator_id is set to NULL
- When a USER (creator) is deleted → USER_STORY.creator_id is set to NULL

4.2 Data Integrity

UNIQUE Constraints:

- USERS.email must be unique
- USERS.username must be unique

NOT NULL Constraints:

- All primary keys
- User credentials (email, username, password)
- Task.project_id (task must belong to a project)
- Comment.task_id and Comment.author_id

ENUM Constraints:

- USERS.role: 'Admin', 'Manager', 'Developer'
- TASKS.status: 'To Do', 'In Progress', 'Done'

5. Indexes for Performance

5.1 Primary Indexes

All primary keys have clustered indexes

5.2 Secondary Indexes

- USERS.email (for login queries)
- USERS.username (for login queries)
- PROJECTS.name (for search)
- TASKS.title (for search)
- TASKS.status (for filtering)

5.3 Foreign Key Indexes

All foreign keys are automatically indexed for join performance

6. Sample Queries

6.1 Get all tasks for a project with assignee details

```
SELECT t.*, u.full_name as assignee_name, u.role
FROM tasks t
LEFT JOIN users u ON t.assignee_id = u.id
WHERE t.project_id = ?
ORDER BY t.deadline ASC;
```

6.2 Get project statistics

```
SELECT
    p.id,
    p.name,
    COUNT(t.id) as total_tasks,
    SUM(CASE WHEN t.status = 'Done' THEN 1 ELSE 0 END) as completed_tasks,
    SUM(CASE WHEN t.status = 'In Progress' THEN 1 ELSE 0 END) as
in_progress_tasks,
    SUM(CASE WHEN t.deadline < NOW() AND t.status != 'Done' THEN 1 ELSE 0 END)
as overdue_tasks
FROM projects p
LEFT JOIN tasks t ON p.id = t.project_id
GROUP BY p.id, p.name;
```

6.3 Get user's assigned tasks

```
SELECT t.*, p.name as project_name
FROM tasks t
JOIN projects p ON t.project_id = p.id
WHERE t.assignee_id = ?
ORDER BY t.deadline ASC;
```

7. Database Normalization

This schema follows **Third Normal Form (3NF)**:

1. **First Normal Form (1NF):** All attributes contain atomic values
2. **Second Normal Form (2NF):** No partial dependencies on composite keys
3. **Third Normal Form (3NF):** No transitive dependencies

7.1 Benefits

- Minimal data redundancy
- Data integrity maintained
- Easy to update and maintain
- Efficient queries with proper indexing

8. Migration Strategy

8.1 Initial Setup

Create tables in the following order (respecting foreign key dependencies):

1. CREATE TABLE users
2. CREATE TABLE projects
3. CREATE TABLE project_members
4. CREATE TABLE tasks
5. CREATE TABLE comments
6. CREATE TABLE user_stories

8.2 Adding Indexes

Add indexes after table creation for better performance:

```
-- Add indexes after table creation for better performance
CREATE INDEX idx_users_email ON users(email);
CREATE INDEX idx_users_username ON users(username);
CREATE INDEX idx_projects_name ON projects(name);
CREATE INDEX idx_tasks_status ON tasks(status);
```

Note: This ER diagram represents the complete database schema for the Project Management Tool. All relationships and constraints are implemented using SQLAlchemy ORM in the backend.

Entity Relationship Diagram | Database Schema for Project Management Tool

Generated on October 26, 2025