# PlanItRight - Event Management System

## Database Schema

Prepared By:
Vivekanand. R
Kiran Kumar Reddy. R

September 3, 2024

## Contents

# 1 Introduction

The database schema is a fundamental component of the PlanItRight Event Management System, providing the structural foundation needed to store, retrieve, and manage data efficiently. This document outlines the comprehensive database schema that underpins the system, detailing the tables, columns, and relationships necessary to support the various functionalities of the platform.

The database schema is designed to ensure data integrity, scalability, and performance while accommodating the diverse requirements of event management. It addresses the need for robust data storage solutions that can handle complex relationships between entities such as users, events, guests, tasks, vendors, payments, and budgets.

## 1.1 Objectives of the Database Schema

The primary objectives of the PlanItRight database schema include:

- **Data Integrity:** Ensuring that all data stored in the database is accurate, consistent, and reliable. This is achieved through the use of primary keys, foreign keys, and constraints that enforce the correct relationships between tables.
- **Scalability:** Designing the database to handle increasing amounts of data and higher loads as the number of events, users, and associated data grows. The schema supports efficient querying and data retrieval to maintain performance even under heavy usage.
- **Security:** Protecting sensitive data through the implementation of appropriate access controls and encryption techniques. The schema is structured to minimize the risk of unauthorized access and data breaches.
- **Performance Optimization:** Structuring the database to optimize performance for both read and write operations. This includes indexing frequently accessed columns and designing queries to minimize load times.
- **Flexibility and Extensibility:** Allowing for future enhancements and changes to the system without requiring significant alterations to the database schema. This flexibility ensures that the system can evolve with changing business needs.

## 1.2 Overview of the Schema Structure

The schema is organized into several key entities, each representing a critical component of the event management process. These entities are interrelated to reflect the real-world relationships between different aspects of event planning and execution. The core entities include:

- **Users:** Manages user information, roles, and authentication credentials. This table is central to controlling access to the system's features and data.
- **Events:** Stores all information related to events, including details such as name, date, location, and the associated organizer.
- **Guests:** Tracks the guests invited to events, their RSVP statuses, and any group affiliations they may have.

- **Tasks:** Contains details of tasks associated with events, including their assignments, status, and deadlines.
- **Vendors:** Manages vendor information, including the services they provide and their association with specific events.
- **Payments:** Logs payments made to vendors, including amounts, payment dates, and statuses.
- **Budgets:** Handles budgeting information for events, tracking total allocated funds, expenditures, and remaining budgets.

### 1.3 Key Considerations in Schema Design

When designing the PlanItRight database schema, several critical considerations were taken into account:

- **Normalization:** The schema is normalized to reduce redundancy and ensure that data is stored efficiently. This approach minimizes the risk of data anomalies and improves consistency across the database.
- **Relationship Management:** The schema carefully defines relationships between entities, such as one-to-many and many-to-one associations, to accurately model the interactions between different components of the system.
- **Data Consistency:** Integrity constraints such as foreign keys are used to maintain data consistency across related tables. This ensures that references between tables remain valid and that the data reflects real-world entities and relationships.
- **Query Efficiency:** Indexes are implemented on frequently queried columns to enhance the speed of data retrieval operations. This is particularly important for large datasets where performance can be a critical factor.

## 2 Database Schema

### 2.1 Tables and Relationships

The following tables are included in the schema:

### 2.2 Users Table

| **Column Name** | **Data Type** | **Constraints** | **Description** |
|---|---|---|---|
| user_id | INT | PRIMARY KEY, AUTO_INCREMENT | Unique identifier for each user |
| username | VARCHAR(50) | UNIQUE, NOT NULL | User's login name |
| password | VARCHAR(255) | NOT NULL | User's hashed password |
| email | VARCHAR(100) | UNIQUE, NOT NULL | User's email address |

## 2.3 Events Table

| **Column Name** | **Data Type** | **Constraints** | **Description** |
|---|---|---|---|
| event_id | INT | PRIMARY KEY, AUTO_INCREMENT | Unique identifier for each event |
| event_name | VARCHAR(100) | NOT NULL | Name of the event |
| event_description | TEXT | | Description of the event |
| event_date | DATE | NOT NULL | Date of the event |
| event_time | TIME | | Time of the event |
| location | VARCHAR(255) | NOT NULL | Location of the event |

## 2.4 Guests Table

| **Column Name** | **Data Type** | **Constraints** | **Description** |
|---|---|---|---|
| guest_id | INT | PRIMARY KEY, AUTO_INCREMENT | Unique identifier for each guest |
| event_id | INT | FOREIGN KEY (event_id) REFERENCES Events(event_id) | Associated event ID |
| guest_name | VARCHAR(100) | NOT NULL | Name of the guest |
| email | VARCHAR(100) | NOT NULL | Email address of the guest |
| phone_number | VARCHAR(20) | | Phone number of the guest |
| rsvp_status | ENUM('Yes', 'No', 'Maybe') | DEFAULT 'Maybe' | RSVP status of the guest |

## 2.5 Tasks Table

| **Column Name** | **Data Type** | **Constraints** | **Description** |
|---|---|---|---|
| task_id | INT | PRIMARY KEY, AUTO_INCREMENT | Unique identifier for each task |
| event_id | INT | FOREIGN KEY (event_id) REFERENCES Events(event_id) | Associated event ID |
| task_name | VARCHAR(100) | NOT NULL | Name of the task |
| task_description | TEXT | | Description of the task |
| assigned_to | INT | FOREIGN KEY (user_id) REFERENCES Users(user_id) | User assigned to the task |
| due_date | DATE | | Due date for the task |
| status | ENUM('To Do', 'In Progress', 'Completed') | DEFAULT 'To Do' | Status of the task |
| priority | ENUM('High', 'Medium', 'Low') | DEFAULT 'Medium' | Priority level of the task |

## 2.6 Vendors Table

| Column Name | Data Type | Constraints | Description |
|---|---|---|---|
| vendor_id | INT | PRIMARY KEY, AUTO_INCREMENT | Unique identifier for each vendor |
| vendor_name | VARCHAR(100) | NOT NULL | Name of the vendor |
| contact_info | VARCHAR(255) | NOT NULL | Contact information of the vendor |
| service_type | VARCHAR(100) | NOT NULL | Type of service provided by the vendor (e.g., Catering, Photography) |
| contract_details | TEXT | | Details of the contract with the vendor |
| event_id | INT | FOREIGN KEY (event_id) REFERENCES Events(event_id) | Associated event ID |

## 2.7 Payments Table

| Column Name | Data Type | Constraints | Description |
|---|---|---|---|
| payment_id | INT | PRIMARY KEY, AUTO_INCREMENT | Unique identifier for each payment |
| vendor_id | INT | FOREIGN KEY (vendor_id) REFERENCES Vendors(vendor_id) | Associated vendor ID |
| event_id | INT | FOREIGN KEY (event_id) REFERENCES Events(event_id) | Associated event ID |
| amount | DECIMAL(10, 2) | NOT NULL | Payment amount |
| payment_date | DATE | NOT NULL | Date of payment |
| status | ENUM('Pending', 'Completed') | DEFAULT 'Pending' | Payment status |

## 2.8 Budgets Table

| Column Name | Data Type | Constraints | Description |
|---|---|---|---|
| budget_id | INT | PRIMARY KEY, AUTO_INCREMENT | Unique identifier for each budget |
| event_id | INT | FOREIGN KEY (event_id) REFERENCES Events(event_id) | Associated event ID |
| total_budget | DECIMAL(12, 2) | NOT NULL | Total allocated budget for the event |
| amount_spent | DECIMAL(12, 2) | NOT NULL | Amount spent so far |

| | | | |
|---|---|---|---|
| remaining_budget | DECIMAL(12, 2) | GENERATED ALWAYS AS (total_budget - amount_spent) STORED | Remaining budget |

# 3 Relationships

The database schema includes the following relationships:

- A **User** can create multiple **Events** (One-to-Many).
- An **Event** can have multiple **Guests**, **Tasks**, **Vendors**, **Payments**, and a single **Budget** (One-to-Many for Guests, Tasks, Vendors, Payments; One-to-One for Budget).
- A **Vendor** can provide services for multiple **Events** (One-to-Many).
- A **Payment** is associated with a specific **Vendor** and **Event** (Many-to-One).

# 4 Conclusion

This database schema for the PlanItRight Event Management System is designed to effectively manage event-related data, including users, events, guests, tasks, vendors, payments, and budgets. The schema ensures data integrity and supports the system's functional and non-functional requirements, enabling efficient and scalable event management.