Party Crafter Database Design

Business Requirements

Summary

Party Crafter is a comprehensive party planning tool that empowers users to effortlessly create, organize, and manage their events. From customizing invitations to tracking budgets and managing guest lists, Party Crafter simplifies the entire party planning process, ensuring memorable celebrations. This project aims to create an efficient database system that will store and manage related user and party data to support Party Crafter.

Objective:

To design a database system for a user-friendly web application that empowers individuals to plan, organize, and manage their own parties and events efficiently, providing a comprehensive platform for event management and coordination.

The database will focus on:

- Allowing users to create new events, specifying details such as event type, date, time, location, and theme.
- Enabling users to invite guests, manage RSVPs, and track attendees.
- Allowing users to track and manage vendors, caterers, decorators, and other event service providers for comparison and better decision-making process.
- Allowing users to set budgets and track expenses.
- Assisting users in creating to-do lists for event planning tasks.

Summary of keywords

Nouns: user, event, event type, date, time, location, theme, guest, RSVP, service provider, budget, expense, to-do list, task

Verbs: create, specify, invite, manage, track, set

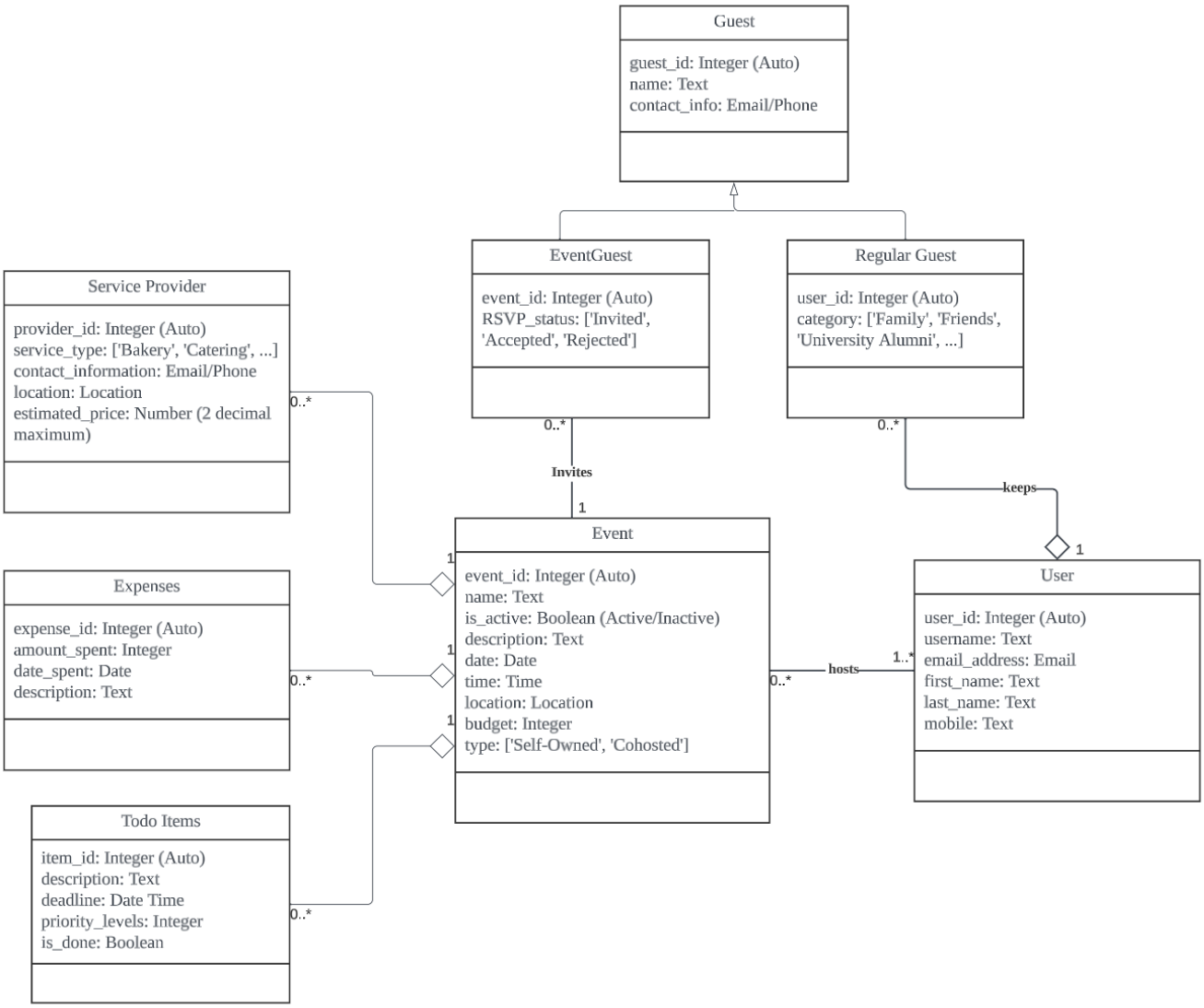
Rules

1. Users should be able to login with selected usernames.
2. Users can create and customize multiple events with event details.
3. Events are divided into 2 types, self-owned and cohost. If a user creates a self-owned event, that event can only be managed and modified by the user created. If a user creates a cohosting event, all users under the cohost list are given the access to that event.
4. Guests are divided into two distinct categories: event guests and regular guests.
5. Event guests are individuals invited to specific events. Each event guest is associated with one and only one event. This association ensures that event hosts can manage their guest lists effectively for each event.
6. Every event within the system may have zero or more event guests. Each event guest is marked with an RSVP status, indicating their response to the event invitation.
7. Regular guests are also guests, but they are not directly associated with events. Instead, each regular guest is linked to one user. This allows users to maintain a list of regular guests

who may be invited to multiple events without the need to re-enter their information for each occasion.

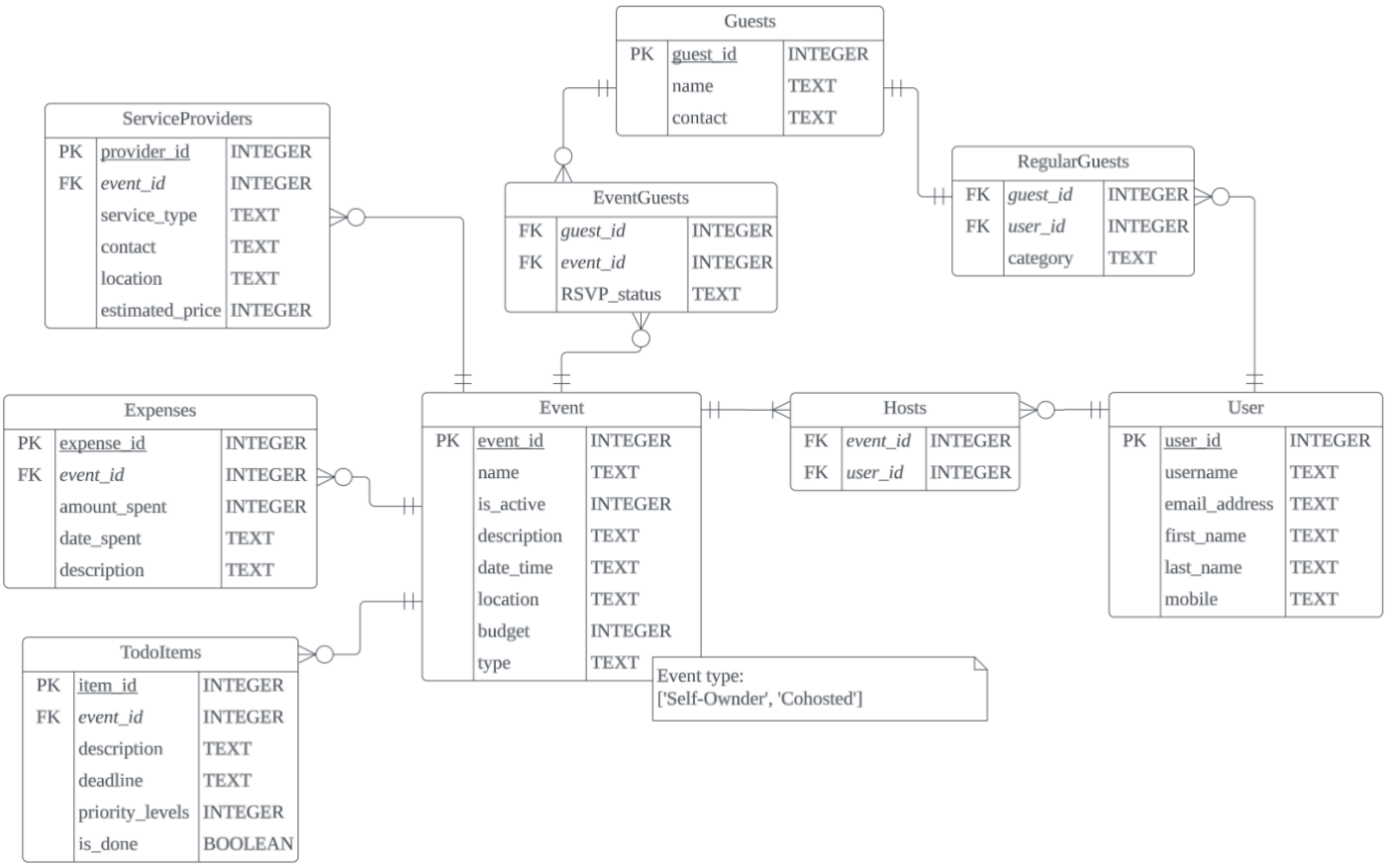
8. Users can create one to-do list for a specific event. Each to-do list contains zero or more to-do items with deadlines and priority levels.
9. Users can create a service provider list for a specific event with details including service type, contact information, location, estimated price, etc.
10. Users can store a budget for a specific event.
11. Users can track and manage expenses for a given event. Each event is associated with zero or more expenses, while each expense is only associated with one single event.

Conceptual Design – UML Diagram



Link: https://lucid.app/lucidchart/28053283-0464-4c91-adb7-bc9c6036d8de/edit?page=0_0&invitationId=inv_a6b3f5d0-525c-4bcf-a6c0-9062424d1d6b#

Logical Data Model - ER Diagram



Link: https://lucid.app/lucidchart/5ee423a2-6159-4431-ba62-2058d444f2ac/edit?page=0_0&invitationId=inv_360d42ed-0fee-411f-886a-4d2a540e44a5#

Schema & Proof

Schema

Users (user_id, username, email_address, first_name, last_name, date_of_birth, mobile)
Events (event_id, name, is_active, event_description, date_time, location, budget, type)
Hosts (*event_id*, *user_id*)
Guests (guest_id, name, contact)
EventGuests (*guest_id*, *event_id*, RSVP_status)
RegularGuests (*guest_id*, *user_id*, category)
TodoItems (item_id, *event_id*, item_description, deadline, priority_levels)
ServiceProvider (provider_id, *event_id*, service_type, contact, location, estimated_price)
Expenses (expense_id, *event_id*, amount_spent, date_spent, expense_description)

Proof of BCNF

Users: No non-trivial functional dependencies other than the candidate key, so it's in BCNF.
Events: No non-trivial functional dependencies other than the candidate key, so it's in BCNF.
Hosts: {event_id, user_id} is the primary key, and there are no non-trivial functional dependencies, so it's in BCNF.
Guests: No non-trivial functional dependencies other than the candidate key, so it's in BCNF.
EventGuests: {guest_id, event_id} is the primary key, and there are no non-trivial functional dependencies, so it's in BCNF.
RegularGuests: {guest_id, user_id} is the primary key, and there are no non-trivial functional dependencies, so it's in BCNF.
TodoItems: No non-trivial functional dependencies other than the candidate key, so it's in BCNF.
ServiceProvider: No non-trivial functional dependencies other than the candidate key, so it's in BCNF.
Expenses: No non-trivial functional dependencies other than the candidate key, so it's in BCNF.