**Project Proposal**

**a) Brief Description of the problem and opportunity**

The hospitality business has experienced an increase in peer-to-peer property rentals in the present digital age. Individuals are looking for innovative, economical, and convenient lodging, while property owners are looking for effective methods to monetize their vacant premises. However, there is a gap in platforms that provide a unified experience for both hosts and guests, while also providing trust, flexibility, and user-centric features. Furthermore, there is a potential to implement

2 user data to improve the booking experience, provide personalized suggestions, and build a community of trustworthy hosts and delighted visitors.

**b) How the proposed Database will address the problem/opportunity**

The database will provide a consistent platform for hosts to list their properties and guests to make bookings. This guarantees that the process is as simple as possible for both sides.

**Property Recommendations:** By collecting visitor preferences and feedback, the platform may provide customized property recommendations, boosting the user experience.

**Transparency and trust:** The feedback and rating system will encourage trust. Guests can select hotels based on previous ratings, and hosts can enhance their services in response to comments.

**Loyalty Programs:** By utilizing the Loyalty Points feature for guests, the platform can implement loyalty programs that provide discounts or rewards to repeat users.

**Superhost Recognition:** Using the SuperhostStatusID, the platform will be able to recognize and reward top-performing hosts, encouraging them to keep up their high standards.

**Dynamic Pricing:** The PricePerNight attribute can be changed based on occupancy rate, allowing hosts to create revenue-maximizing dynamic pricing methods.

**Secure Transactions:** The Payment entity makes certain that all transactions are recorded, resulting in a secure and transparent payment process.

**Entities, Attributes, and Relationships**

**User (Superclass):**

* **Attributes:** UserID, Username, Email, DateofJoining, PhoneNumbers.
* **Relationships:** A User can be either a Guest or a Host, but not both.
* **Justification:** The User entity acts as a superclass to capture common attributes and behaviors of both Guests and Hosts. This design promotes data integrity and reduces redundancy.

**a) Guest (Subclass of User):**

* **Attributes:** Preferences, Loyalty Points.
* **Relationships:** A Guest can make multiple Reservations, make payments, and can give a Guest Review.
* **Justification:** By creating a Guest subclass, specific attributes and behaviors related to guests are captured.

**b) Host (Subclass of User):**

* **Attributes:** SuperhostStatus.
* **Relationships:** A Host can list multiple listings/properties and post a HostReview.
* **Justification:** The Host subclass captures attributes and behaviors specific to hosts. The SuperhostStatus can be used to offer special privileges or promotions to superhosts.

**Listing / Property:**

* **Attributes:** PropertyID, Address, RoomType.
* **Relationships:** A property is owned by one Host and each property will have Media.
* **Justification:** The property entity captures all details about properties available for reservation. The UserID as a foreign key ensures that each listing is associated with a host.

**Reservation:**

* **Attributes:** ReservationID, StartDate, EndDate, TotalPrice, Cancellation, DateofBooking.
* **Relationships:** A Reservation is an agreement between a Guest and a Listing.
* **Justification:** The Reservation entity captures the details of a booking. By linking it with UserID and PropertyID, it ensures that each reservation is associated with a guest and a property.

**HostReviews:**

* **Attributes:** HostReviewID, HostRating
* **Justification:** The Host Reviews entity captures feedback specifically about the guest's behavior during their stay.

**GuestReviews:**

* **Attributes:** GuestReviewID, GuestRating
* **Justification:** The GuestReviews entity allows guests to provide ratings for listings, which can be useful for future guests and for hosts to improve their services.

**Payments:**

* **Attributes:** PaymentID, PaymentDate, Price
* **Justification:** The Payments entity captures transaction details. By linking it with ReservationID and UserID, it ensures that each payment is associated with a reservation.

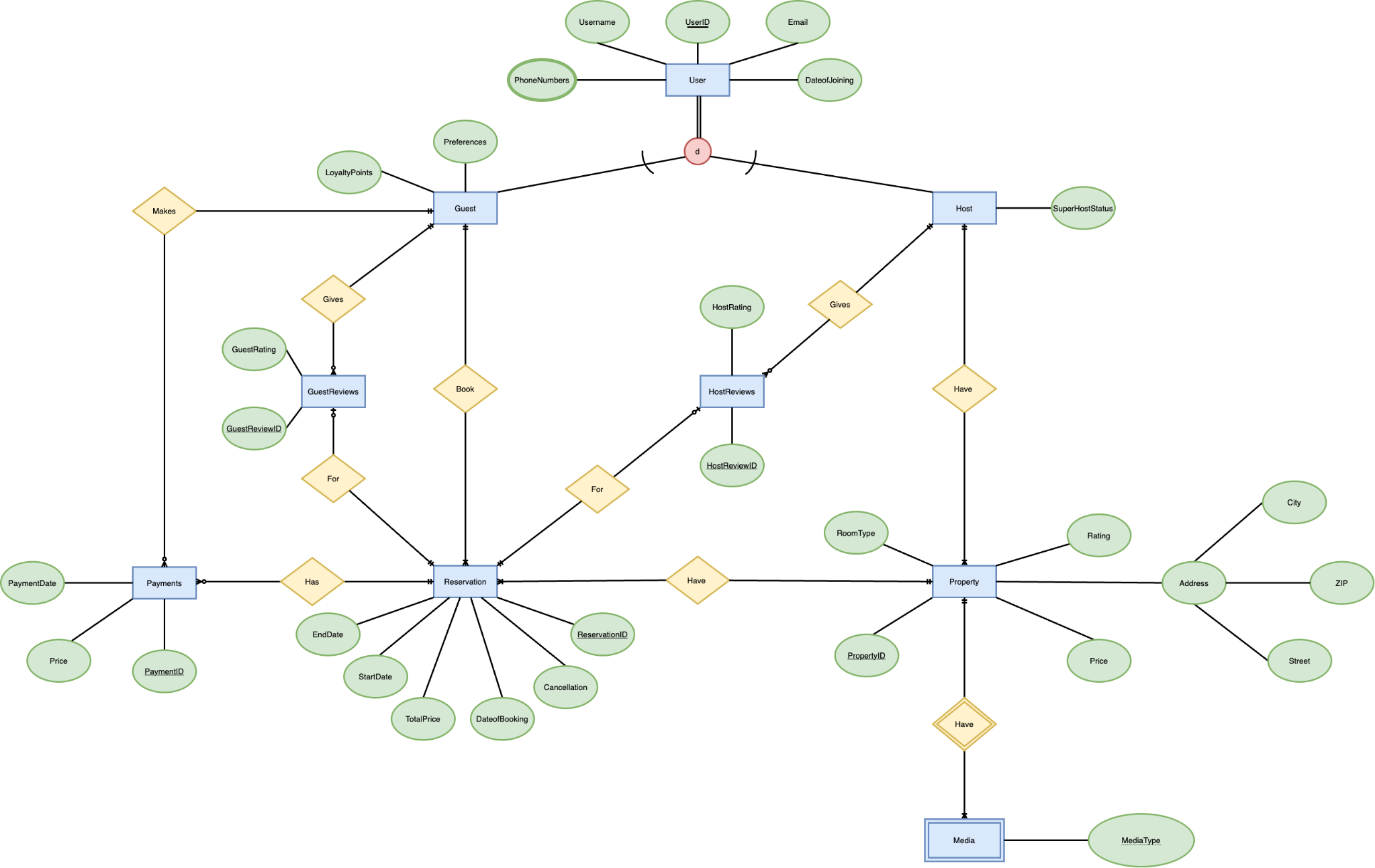
**Media:**

* **Attributes:** Mediatype
* **Justification:** The media entity is a weak entity because it cannot exist without property/listing. It contains photos/videos of the property.

**Project Justification:**

* **Data Integrity:** The design ensures that data is consistent and accurate. For example, a reservation cannot exist without a guest and a listing.
* **Flexibility:** The design allows for easy expansion. New attributes or entities can be added without major changes to the existing structure.
* **User Experience:** By capturing feedback and ratings, the platform can offer better recommendations to future guests.
* **Business Insights:** The design allows for easy analysis. For example, analyzing the feedback can provide insights into the most popular listings or areas of improvement.
* **Operational Efficiency:** With clear relationships and attributes, operations like booking, payment, and feedback can be streamlined.

In conclusion, this design provides a comprehensive structure for a property booking platform, ensuring data integrity, flexibility, and enhanced user experience. It lays a strong foundation for building a robust and scalable system.

****