COURSE: CS/DSA- 4513 – DATABASE MANAGEMENT
SECTION: 001
SEMESTER: FALL 2019
INSTRUCTOR: DR. LE GRUENWALD

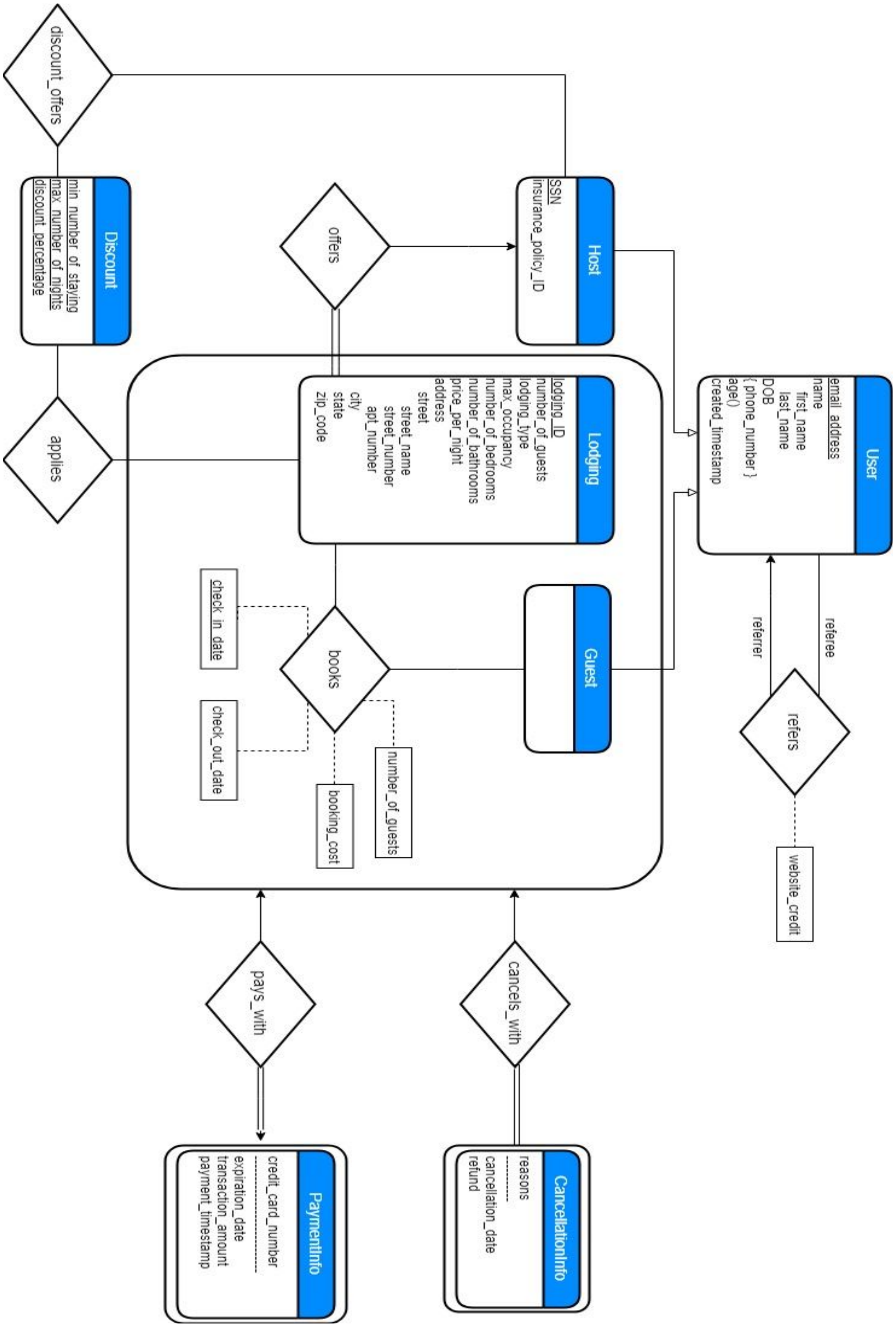
GROUP NUMBER: 43

GROUP MEMBERS:

MASON DRUCTOR
LINCE RUMAINUM
MASON SCHMIDT
TUAN VU

SCORE:

1-1)


discount_offers
Discount
min_number_of_staying
max_number_of_nights
discount_percentage
Host
SSN
insurance_policy_ID
offers
User
email_address
name
first_name
last_name
DOB
{ phone_number }
age()
created_timestamp
Lodging
lodging_ID
number_of_guests
lodging_type
max_occupancy
number_of_bedrooms
number_of_bathrooms
price_per_night
address
street
street_name
street_number
apt_number
city
state
zip_code
applies
Guest
books
check_in_date
check_out_date
booking_cost
number_of_guests
referrer
referee
refers
website_credit
pays_with
cancels_with
PaymentInfo
credit_card_number
expiration_date
transaction_amount
payment_timestamp
CancellationInfo
reasons
cancellation_date
refund

1-2)

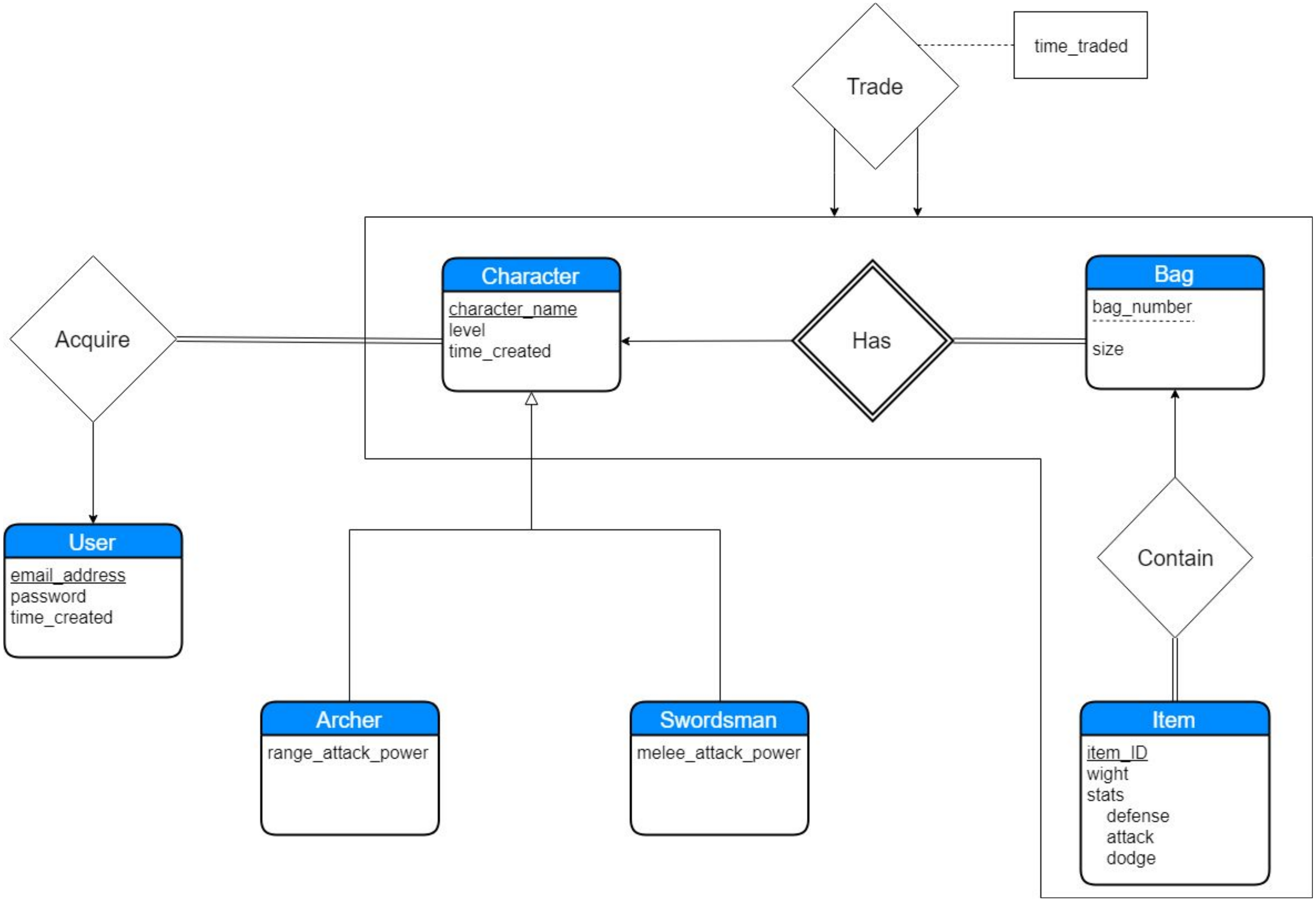
In order to express this in our current design, we would need to add a ternary relationship called “reviews” between a host, a booking, and a new entity, “guest review.” This relationship involves one host with one booking, but could involve the same guest multiple times. We would also need to add that a weak entity relationship “has” to guest and their respective guest reviews. This is a one to many relationship.

We noticed that due to our original design and the addition of this constraint, our new design has a byproduct of additional complexity with a ternary relationship involving an aggregate, as well as another binary relationship. We couldn’t find a simpler design to retrofit into our already designed database, but if designed with this constraint in mind from the beginning, it may have been possible to simplify the complexity of the entity relationships.

2)

A game company needs to create a database to store account related and game world data for their new online action role-playing game “Battlecraft Royale.” The game company wants players to be able to create different types of characters and interact with other players in the game world in various ways.

a) When a user creates a new account for the game, they still need to supply their e-mail address, and create a password which will be stored. The e-mail address must be unique across all accounts. A time-stamp of when the account was created should also be stored.
b) After a user has acquired an account they can create an arbitrary amount of different characters that are associated with the account. These characters will each have their own unique name and a numeric level. A time-stamp should also be generated on character creation to keep track of how long a character has been played.
c) Each character can specialize into one of two classes, an archer or a swordsman. Archers need to store data for Ranged Attack Power, while swordsman need to store data for Melee Attack Power.
d) Each character has its own bag(s) which have a variable size that must be stored. A bag number is used to distinguish bags belonging to the same user. Each bag can be filled with items, which each item having a unique item ID, a weight, and a set of stats composed of defense, attack, and dodge.
e) Characters should be able to trade items with other characters in the game world. This lets two characters access their bags and swap items with each other. A time-stamp of the trade attempt should be stored for authentication.


Trade
time_traded
Acquire
Character
character_name
level
time_created
Has
Bag
bag_number
size
User
email_address
password
time_created
Contain
Archer
range_attack_power
Swordsman
melee_attack_power
Item
item_ID
wight
stats
defense
attack
dodge