The CUSTOMER entity has a user login which uniquely identifies the customer and serves as a login username, a password, and a phone number.

A RESERVATION, which is made by the customer, includes a date and status (“active” or “cancelled”). This table also includes foreign keys for the user that booked the reservation, the event to where the user will be attending, and the GARAGE to which the user will be parking. A combination of event, user ID, and date can be concocted to make the unique confirmation number.

The GARAGE entity includes the name and address of the garage as well as a unique identifier and the number of spaces. The number of spaces can be compared to a count of reservations for a given garage and date to find the availability of the garage.

The EVENT entity includes the name, start date, and end date. EVENT has a foreign key for VENUE, where more information about the venue for the event is stored. The start date and end date also cause an error to be sent when the user attempts to make a reservation outside the attempted date range.

The CANCELLATION entity is a weak entity that references EVENT. If a given day of the EVENT is cancelled, the RESERVATIONs for the event on that day are marked “cancelled” and users can no longer book RESERVATIONs on that day.

VENUE includes a name and address of venues used by EVENTs in the database.

Given that the fee is a function of the garage and the event, the FEE entity is a weak entity that stores the respective fee for every combination of EVENT and GARAGE in the database, which are referenced by foreign keys. When the user makes a reservation, the EVENT and GARAGE that they select will allow the fee to be queried.

To model distance, a DISTANCE table is created with each combination of GARAGE and VENUE, which allows the user view to be ordered by distances when evaluating different parking options.