Diagram

Description automatically generated

The two entity relationship diagrams above, or ERDs, are for the “Cancel Cab” and “Rate Experience” use cases for the potential RazorCab System. The first ERD is for the “Cancel Cab” use case. As notated in the diagram, a customer can have zero to one reservation, while a reservation has one and only one customer. A dispatcher can have zero to many reservations that they are/will be handling, while a reservation has one and only one dispatcher.

The second ERD is for the “Rate Experience” use case in the RazorCab System. Again, a customer can have zero to one reservation while a reservation has one and only one customer. A dispatcher can have zero to many reservations that they may be handling, while a reservation has one and only one dispatcher. The new class in this diagram is the survey. Here, a reservation can have zero to one survey, as the customer may choose not to fill out the survey for their ride. However, each survey can have one and only one reservation linked to it.

Diagram

Description automatically generated

0..

The two diagrams above are the Class Diagrams for the “Cancel Cab” and “Rate Experience” use cases for the RazorCab System. The diagrams are very similar to the ERD’s for these two use cases, but with different notation. For the “Cancel Cab” use case, a customer can have zero to one reservations in their name, while a reservation only has one customer. A dispatcher can have zero to many reservations that they may be handling while a reservation is handled by one and only one dispatcher. The same goes for the “Rate Experience” use case, but with the added component of the survey. A reservation can have zero to one surveys, as a customer may choose to not fill out a survey for their reservation, while a survey has one and only one reservation linked to it.

**Summation for Class Diagrams**

With these class diagrams, our talented team of developers will be able to construct functions that meet each use case’s needs accordingly. They will make sure that each class object has the necessary attributes to work together accordingly. They will also be able to transfer this data from the application into and out of the database system. For example, each survey that is submitted will be stored into the database, and the marketing and customer service teams will be able to view the stored surveys at any given time. We have full confidence in our team of developers that they will meet the required use case functionality and attributes from these diagrams. We cannot wait to being working on the RazorCab System!