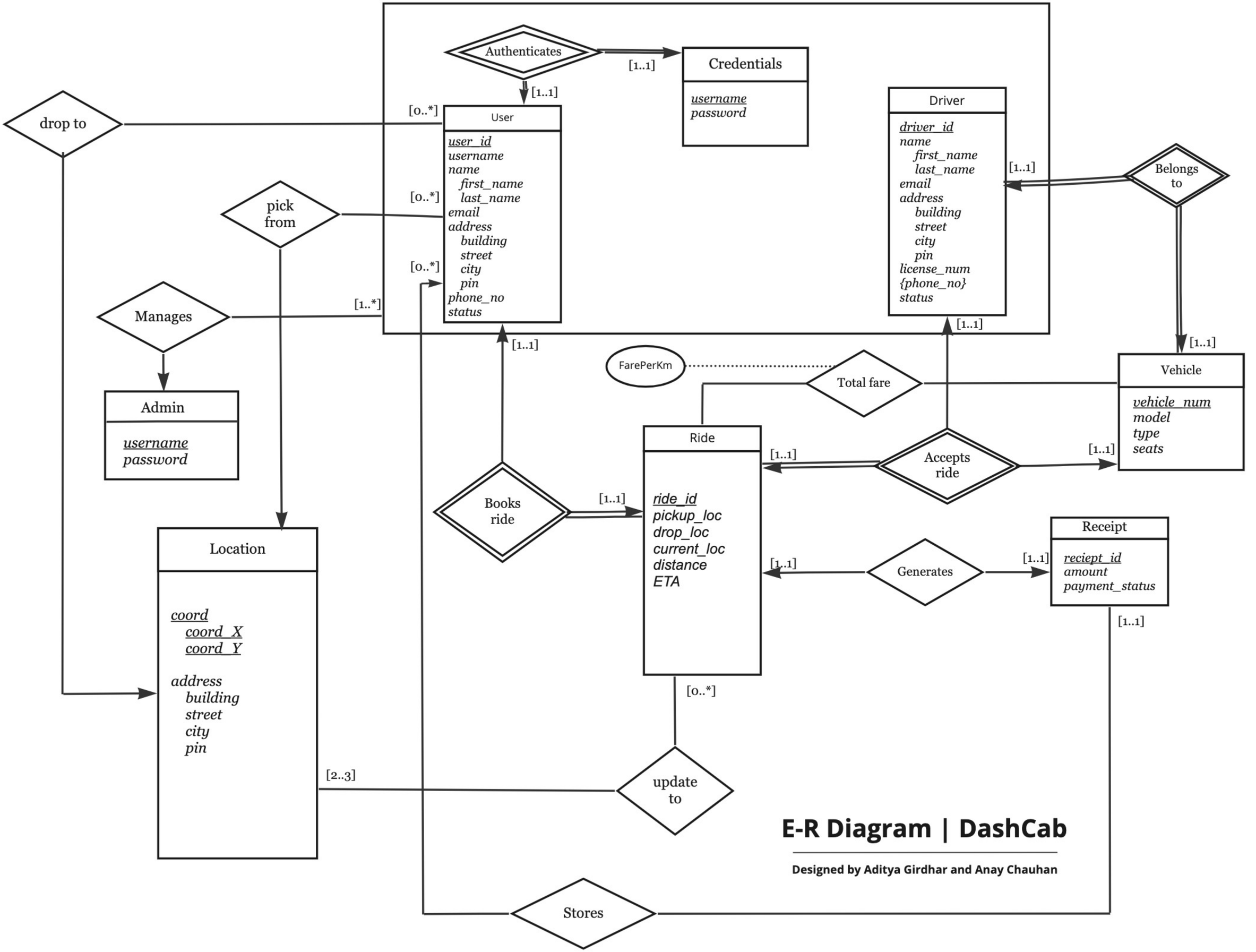
Authenticates
[1..1]
Credentials
username
password
[1..1]
drop to
[0..*]
User
user_id
username
name
first_name
last_name
email
address
building
street
city
pin
phone_no
status
Driver
driver_id
name
first_name
last_name
email
address
building
street
city
pin
license_num
{phone_no}
status
[1..1]
Belongs to
[1..1]
pick from
[0..*]
[0..*]
[0..*]
Manages
[1..*]
[1..1]
[1..1]
Admin
username
password
FarePerKm
Total fare
Vehicle
vehicle_num
model
type
seats
Ride
ride_id
pickup_loc
drop_loc
current_loc
distance
ETA
[1..1]
Accepts ride
[1..1]
Books ride
[1..1]
[1..1]
Generates
[1..1]
Receipt
reciept_id
amount
payment_status
[1..1]
Location
coord
coord_X
coord_Y
address
building
street
city
pin
[0..*]
[2..3]
update to
Stores
E-R Diagram | DashCab
Designed by Aditya Girdhar and Anay Chauhan

User

user_id	username	first_name	last_name	email	building	street	city	status	pin	phone_no

Driver

driver_id	vehicle_num	first_name	last_name	email	building	street	status	city	pin	license_num	{phone_no}

Admin

username	password

Location

coord_X	coord_Y	building	street	city	pin

Ride

ride_id	driver_id	user_id	pickup_loc	drop_loc	current_loc	distance	ETA

Receipt

receipt_id	ride_id	amount	payment_status

Vehicle

vehicle_num	driver_id	model	type	seats

Credentials

username	password

Relational Schema | DashCab

Designed by Aditya Girdhar and Anay Chauhan

Primary key
Foreign key
Attribute

Scope of the Work:

The app has been organized in such a way that a person can use it as a client/customer/user or as a driver. Each user will have a unique user_id, username, name, email id, status, and address(which is a composite attribute containing building, street, city name, and PIN) along with a unique phone number. Similarly, the driver too has a unique driver_Id, a unique name, an address(which is a composite attribute containing building, street, city name, and PIN), an array of phone numbers, status(whether currently in a ride or not), and also the driver's license number. The user is authenticated from the corresponding credentials saved in the database. Both, the user and the driver entities are managed by the admin. Each driver is also associated with a single vehicle and vice-versa. These vehicles have various properties(attributes) such as model, vehicle_num(which is the primary key), and also the type of the vehicle(SUV, sedan, etc). A user books a ride that is accepted by a single driver depending on the distance of the driver from the user. Each ride can be identified by the corresponding username (of the user/client) and the driver_Id. Each driver can accept multiple rides during the day but at a single time will only have a single corresponding ride. The same applies to a user/client as well. Each ride will generate a single receipt(the price of which is calculated using the fare() function on the basis of charge per kilometer). Along with this it also has a unique receipt_id for identification and payment status telling whether the payment was successful or not. All of these receipts are stored by the instance of the user entity to which they correspond. Each ride simultaneously corresponds to 3 locations on the map, the first being the pick-up location, the second being the drop-off location and lastly, the third being the current location.