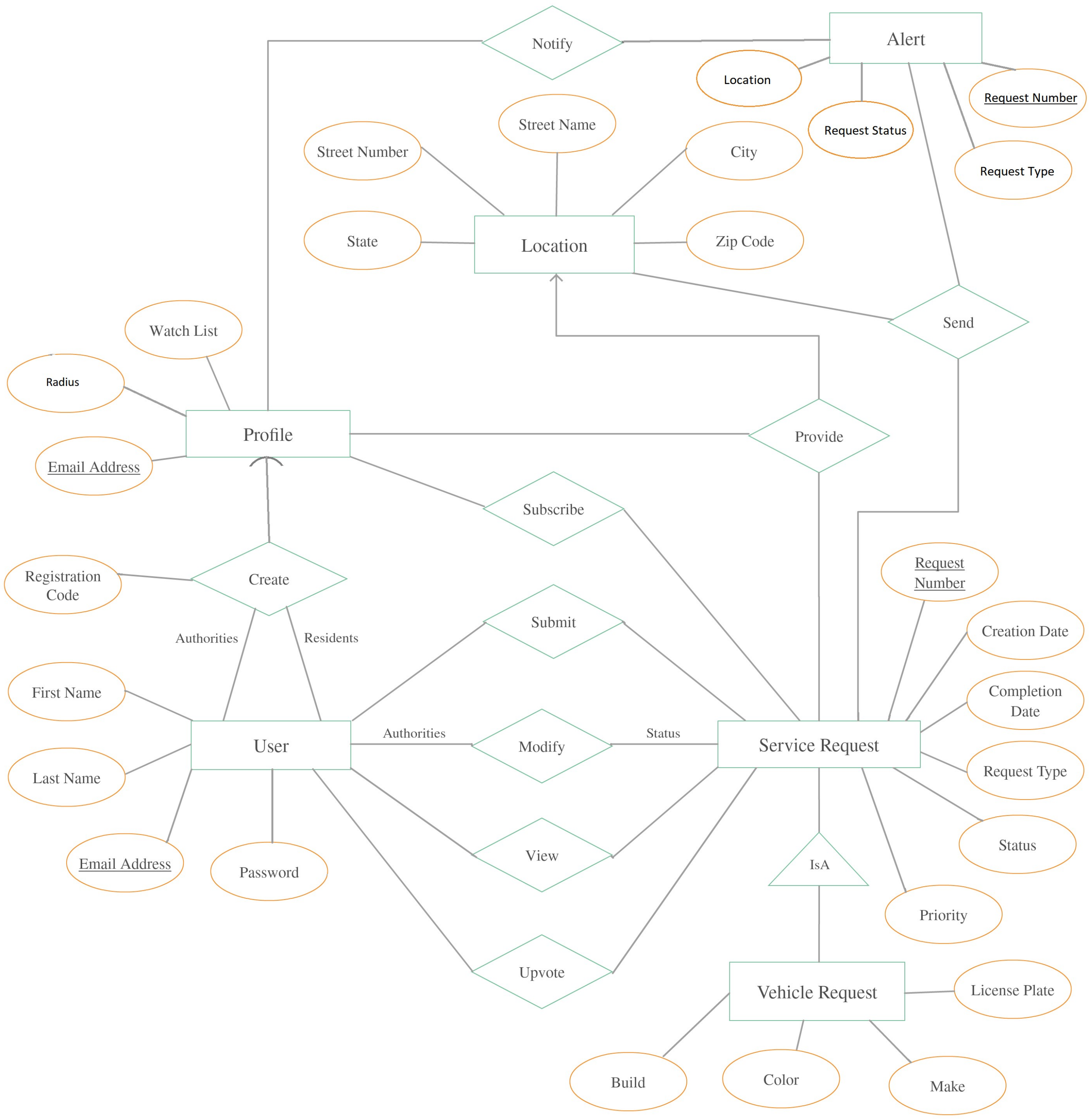
Project HawkEye Stage 1

Team Name	HawkEye
Members and Names (NetIDs)	Alice Yu (aliceyu2), Collin Ryals (cryals2), Someshwar Vaddepally (sv17), Ahtesham Ali (saali4)
Emails	aliceyu2@illinois.edu, cryals2@illinois.edu, sv17@illinois.edu, saali4@illinois.edu
Team Captain	Alice Yu (aliceyu2)
Project Title	Chicago Service Requests (CSR)
Project Summary	Using a dataset from Kaggle containing records of the different service requests created in the year 2018, our database application will provide Chicago residents the ability to report new requests, check the status of existing requests, and be alerted of newly created or completed service requests that have been reported in their area.
Project Description	● **Application description**: Our application will take all 311 Chicago city service requests that were created in the year 2018 and perform different basic and advanced functions on the database to create a user-friendly interface that will help residents track the status of different types of service requests based on user input. CSR will help residents report new requests, view all existing requests, increase the priority of current requests, and stay informed about all newly created and completed service requests in their area via an autonomous notification system. ● **Usefulness**: In contrast to other applications that currently only allow users to check the status of different types of service requests (e.g., https://www.chicago.gov/city/en/depts/311.html), our application will have the ability to display the 20 highest priority service requests that were created within a one mile radius of any user-specified location. By ranking all open service requests by their priorities, government officials will have a better understanding of which service requests are more urgent for current residents. In addition, our application will allow the user to opt-in to an autonomous alert system that will

	notify the user if any new requests have been opened in their area or if the status of any of those nearby service requests have changed. ● **Dataset description**: The dataset comes from a precompiled database from Kaggle called "Chicago 311 Service Requests". Our main dataset consists of records resulting from a filtering of the original precompiled database to show only the service requests created in 2018 involving pot holes and abandoned vehicles. ● **Functionality description**: Our application's basic functions will include operations such as insertions, updates, deletions, and searches and will be used to clean up older service requests that have been marked as completed over 2 years ago from the database, create additional requests, update existing requests, and display only subset of the database based on a specified service request type, the status of a service request, and the user-specified location. It will also allow users to create their profiles and upvote existing issues that they believe to be a priority. Our application's advanced functions will involve the implementation of geofencing, trigger-based push notifications, and complex join and sub-query operations. Geofencing is a feature that uses a device's GPS or longitude/latitude coordinates to define geographical boundaries. It will allow our application to create geographical boundaries that will define an area for which a user is to receive push notifications for. Trigger-based push notifications are push notifications that get created whenever the conditions for a specific trigger has been met. They will allow our application to notify users real-time whenever certain conditions such as the changing of a service request's status are met. Complex join and sub-query operations are operations that are used to combine data from different tables into a single result. They will allow the user to retrieve records from multiple tables and perform additional operations on them to receive a filtered subset of records based on the user's desires. ● **Advanced techniques**: Our application will also leverage more advanced techniques such as transactions, triggers, compound statements, constraints, and views.
ER Design Statements	● A user can only create 1 profile ● A profile can be created by multiple users ● A user can be either a resident or an authority

	● A user can submit multiple service requests ● A service request must provide only 1 location ● Locations can be provided by many different service requests ● A user can make many service requests ● Service requests can be made by many users ● Service requests must contain a unique request number, a status, a creation date, a completion date, a priority number, and a request type ● Locations must contain a street number, a street name, a city, a state, and a zip code ● Service requests can send multiple alerts ● Alerts can be sent by multiple service requests ● Abandoned vehicle requests are a type of service request ● Abandoned vehicle requests can contain information about the vehicle's license plate, color, make, and model ● Each service request must contain an unique request number that does not exist anywhere else ● Accounts can be notified of multiple alerts ● Alerts can notify multiple accounts ● Alerts will contain the following information: a service request number, a service request type, a status, and a location ● A profile must provide only 1 location ● Locations can be provided by multiple profiles ● Locations can send multiple alerts ● Alerts can be sent by multiple locations ● Users can upvote many service requests ● Service requests can be upvoted by multiple users ● Locations can have the same combination of address, city, state, and zip code ● Each user has an unique email address, a first name, a last name, and a password ● Users can have a combination of a first and last name that are not necessarily unique ● Users can view multiple service requests ● Service requests can be viewed by multiple users ● Users can modify multiple service requests ● Service requests can be modified by multiple users ● Many profiles can subscribe to many service requests ● A profile will contain a watch list (a list containing a list of service requests to send alerts for), a desired radius (defaulting to 1 mile), and the user's associated email address

ER Diagram


Notify
Alert
Location
Request Status
Request Number
Request Type
Street Name
Street Number
City
State
Location
Zip Code
Send
Watch List
Radius
Profile
Email Address
Provide
Subscribe
Registration Code
Create
Authorities
Residents
Request Number
Creation Date
Submit
Completion Date
First Name
User
Authorities
Modify
Status
Service Request
Last Name
Request Type
Status
View
Email Address
Password
IsA
Priority
Upvote
Vehicle Request
License Plate
Build
Color
Make