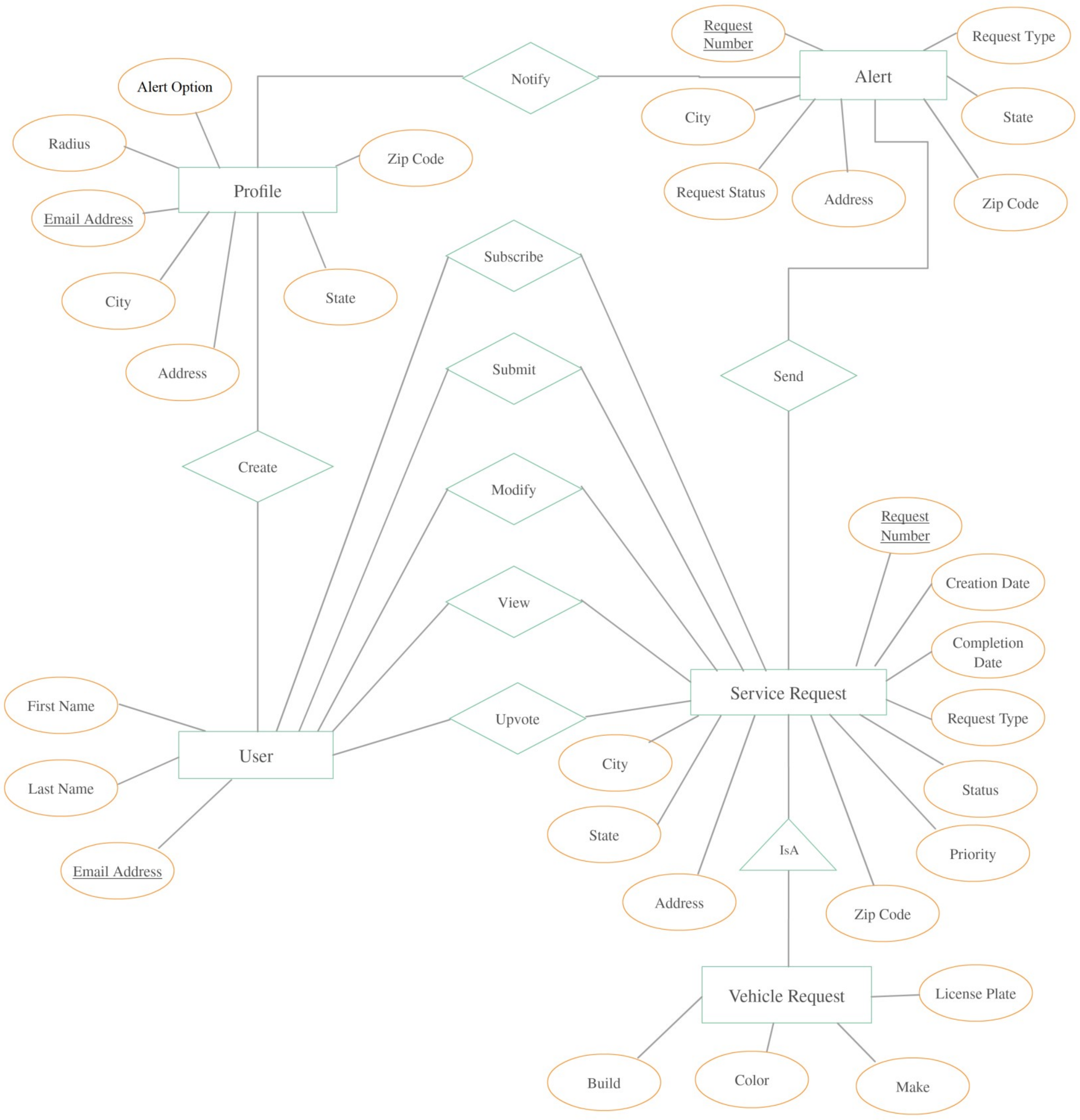
Project HawkEye Stage 4

Team Name	HawkEye
Members and Names (NetIDs)	Alice Yu (aliceyu2), Collin Ryals (cryals2), Ahtesham Ali (saali4)
Emails	aliceyu2@illinois.edu, cryals2@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 years 2018 and 2019, 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 years 2018-2019 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, 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. 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 and 2019 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. 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 ● Each profile can be created by only 1 user ● A user can submit multiple 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, city, state, zip code, address and a request type ● 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 contain a city, state, zip code, and address ● 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 5 mile), and the user’s associated email address

ER Diagram



Request Number
Request Type
Alert
Notify
Alert Option
Radius
Zip Code
Profile
Email Address
City
State
Address
City
State
Request Status
Address
Zip Code
Subscribe
Submit
Send
Create
Modify
Request Number
Creation Date
Completion Date
View
First Name
Upvote
Service Request
Request Type
User
City
Last Name
Status
State
IsA
Priority
Email Address
Address
Zip Code
Vehicle Request
License Plate
Build
Color
Make

Relational Schema	● Profile(Profile.emailAddress, radius, watchList, city, state, zipCode, address) ● User(User.emailAddress, firstName, lastName, password, Profile.emailAddress) ● ServiceRequest(ServiceRequest.requestNumber, creationDate, completionDate, requestType, status, priority, city, state, zipCode, address) ● VehicleRequest(ServiceRequest.requestNumber, build, color, make, licensePlate) ● Alert(Alert.requestNumber, requestType, requestStatus, city, state, zipCode, address) ● Notify(Profile.emailAddress, Alert.requestNumber) ● Send(Alert.requestNumber, ServiceRequest.requestNumber, city, state, zip code, address) ● Subscribe(Profile.emailAddress, ServiceRequest.requestNumber) ● Submit(User.emailAddress, ServiceRequest.requestNumber) ● Modify(authorities_User.emailAddress, status_ServiceRequest.requestNumber) ● View(User.emailAddress, ServiceRequest.requestNumber)
Development Plan	● **Database choice:** MySQL ● **Software platforms:** Django, Apache ● **Software languages:** JavaScript, Python ● **How and where to find data:** Our data will come from a subset of records that originally exists in a precompiled Kaggle database called "Chicago 311 Service Requests". The subset of records that our database will use will come from a filtering of the original database to show only the service requests created in 2018 and 2019 involving pot holes and abandoned vehicles service requests.

Development Plan - Project Timeline and Labor Division

Milestone	Due Date	Milestone Description	Labor Division
1	2/24/19	a. Install all necessary tools, packages, and libraries onto the VMs b. Download all software platforms and languages (if they do not already exist) onto the VMs c. Install and set up the Apache web server onto the VMs d. Set up the backend by creating all of the necessary database tables and populating those tables with the correct columns e. Inject data from the dataset into the database	● Alice: a-d ● Cole: a-e ● Ahtesham: a-c
2	3/3/19	a. Create a FQDN and research how to keep the backend reachable by the front-end at all times b. Create the service request status web page for the website c. Implement a search function that leverages backend operations and reflects the results on the front-end	● Alice: b, c ● Cole: a ● Ahtesham: b
3	3/10/19	a. Design each web page for the web site b. Implement the front-end homepage design for the website c. Implement the login web page for the website d. Implement the registration web page for the website	● Alice: a, b ● Cole: a, d ● Ahtesham: a, c
4	3/17/19	a. Implement the profile web page for the website b. Implement the service request creation web page for the website c. Implement the service request home web page for the website d. Add basic functionality to website (CRUD)	● Alice: a, c, d ● Cole: b, d ● Ahtesham: c, d
5	3/24/19	a. Begin implementing the trigger-based push notification feature (advanced feature #1) b. Begin implementing the geo-fencing feature (advanced feature #2) c. Finalize the decision about which advanced techniques to implement	● Alice: a-c ● Cole: a-c ● Ahtesham: a-c

6	3/31/19	a. Create a demo video that demonstrates CRUD operations (initial video) b. Finish the trigger-based push notification feature (advanced feature #1) c. Finish the geo-fencing feature (advanced feature #2)	● Alice: c ● Cole: b ● Ahtesham: a
7	4/7/19	a. Upload the demo video demonstrating CRUD operations (initial video) b. Implement 1 advanced technique c. Implement 1 advanced technique d. Implement 1 advanced technique	● Alice: a, b ● Cole: c ● Ahtesham: d
8	4/14/19	a. Implement 1 advanced technique b. Implement 1 advanced technique c. Add additional GUI features for better visual appeal	● Alice: a ● Cole: b ● Ahtesham: c
9	4/21/19	a. Perform unit and automation tests b. Add additional GUI features for better visual appeal c. Create a demo video that demonstrates advanced technique operations (final demo)	● Alice: b, c ● Cole: b, c ● Ahtesham: a, c
10	4/28/19	a. Upload the demo video demonstrating advanced function and technique operations (final demo)	● Alice: a

Link to the system's search page:
http://sp19-cs411-12.cs.illinois.edu:8000/serviceRequest/
Note: You must have an account on the website in order to see this page.
Example login information: user1, user1Password

Link to the initial demo:
https://drive.google.com/file/d/1-czzY8bMwHTamUgcISEp1-Yf9CMdXPCb/view?usp=sharing