6.0 Software Design Description

Outline of Software Development Plan

6.1 Introduction

This document presents the architecture and detailed design for the software for CaseX. The Los Angeles Police Department Homicide Library Unit seeks a database solution for the management of case records dating back to 1960. Case records are currently stored in physical binders. Detectives need a more efficient system to store metadata about the case records, query data, and generate reports. Through CaseX's three main functions: Upload Data, Explore Data, and Generate Reports, users are enabled to effectively manage and understand case data.

6.1.1 System Objectives

This system will be uniquely customized to suit the needs of The Los Angeles Police Department Homicide Library Unit. It will allow the staff to more efficiently store and search case data in order to generate reports. Specifically, the system will eliminate the need to spend human resources on physical document retrieval, allow cases to be accessed concurrently from several users, and enable instant, specific queries.

6.1.2 Hardware, Software, and Human Interfaces

Interface Type	Interface Description
Human Interface	Mouse and Keyboard
Human Interface	Monitor
Hardware Interface	Wireless Networking
Software Interface	Database
Software Interface	Server

6.2 Architectural Design

The CaseX system architecture is comprised of a web browser-based user interface (front-end), a server (backend), and MongoDB as the database. The frontend consists of various webpages which allows users to accomplish the three main functions of CaseX. The server backend and database will be designed to allow for concurrent access, efficient query, and data validation.

6.2.1 Major Software Components

- **Front End CSC**
    - About Page
        - Login module
    - Login Page
        - Login input
        - Login submit button
    - Header
        - Menu
        - User info
            - Username
            - Permissions

        - Current page
        - Logout button
    - Home Dashboard Page
        - Links
    - Input Form Page
        - Form select
        - Manual input
        - Form upload
        - Submit
    - Data Explorer Page
        - Query module
            - Search conditions
        - Table display
            - Display filter
            - Table
        - Graphic display
    - Case Page
        - Case selector
        - Display selector
        - Toggle edit
        - Data display
    - Admin Console Page
        - Users management
        - Case database management
- **Server CSC**
    - Queries
        - Create / Get / Edit / Delete Case(s)
        - Create / Get / Edit / Delete Suspect(s)
        - Create / Get / Edit / Delete Victim(s)
        - Create / Get / Edit / Delete Users(s)
    - Routes
        - API Routes
        - View Routes
- **Database CSC**
    - Collections
        - Cases
        - Suspects
        - Users
        - Victims

6.2.2 Major Software Interactions

The three main software components for CaseX are the web client, server, and database. When a user navigates to CaseX URL endpoints, the server responds with an html page for the web client to render. Through the user interface, the web client is able to create, read, update, and delete data by making jQuery AJAX calls to the RESTful API provided by the server. The server uses mongoose object relational mapping to perform data validation of the payload and establishes a connection to the MongoDB database. The database returns data and a status code, which the server then parses before sending the data back to the web client.

6.2.3 Architectural Design Diagrams

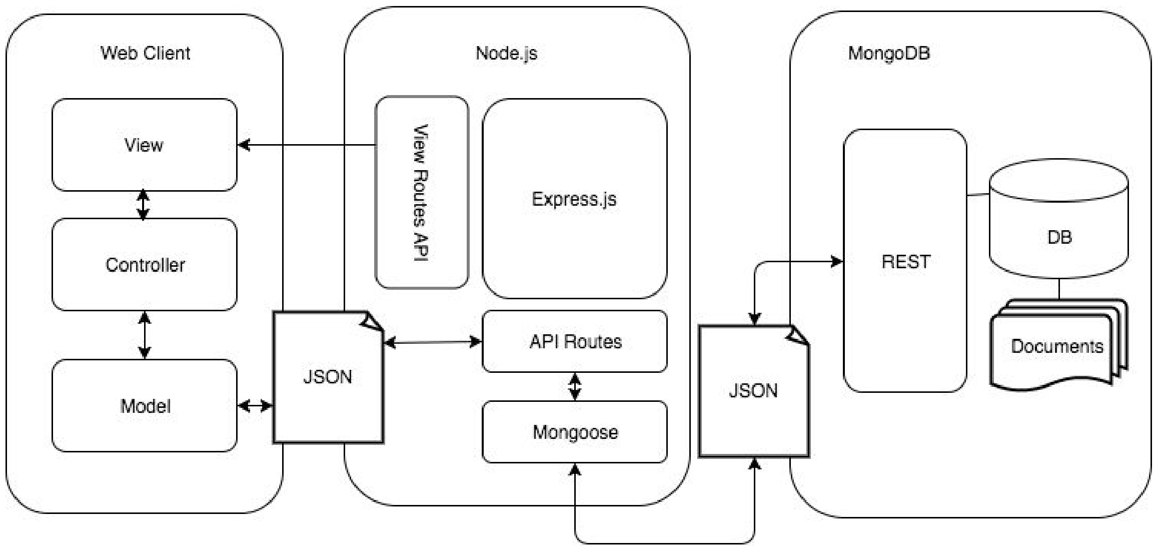


Figure 1: CaseX Software Architectural Diagram

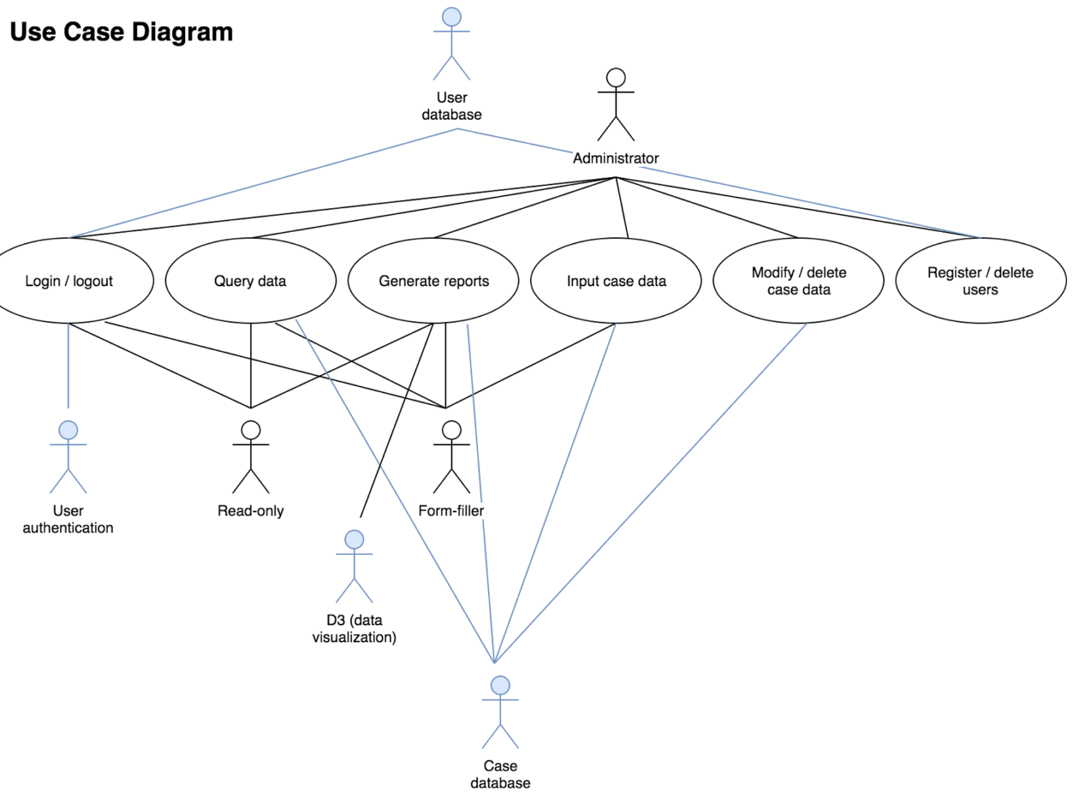


Figure 2: CaseX Use Case Diagram for Overall Features

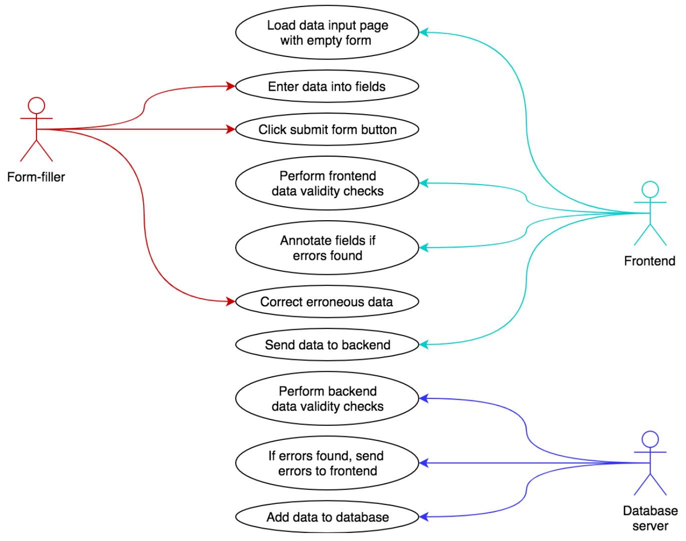

Load data input page with empty form
Enter data into fields
Click submit form button
Perform frontend data validity checks
Annotate fields if errors found
Correct erroneous data
Send data to backend
Perform backend data validity checks
If errors found, send errors to frontend
Add data to database
Form-filler
Frontend
Database server

Figure 3: Use Case Diagram for Submitting Data

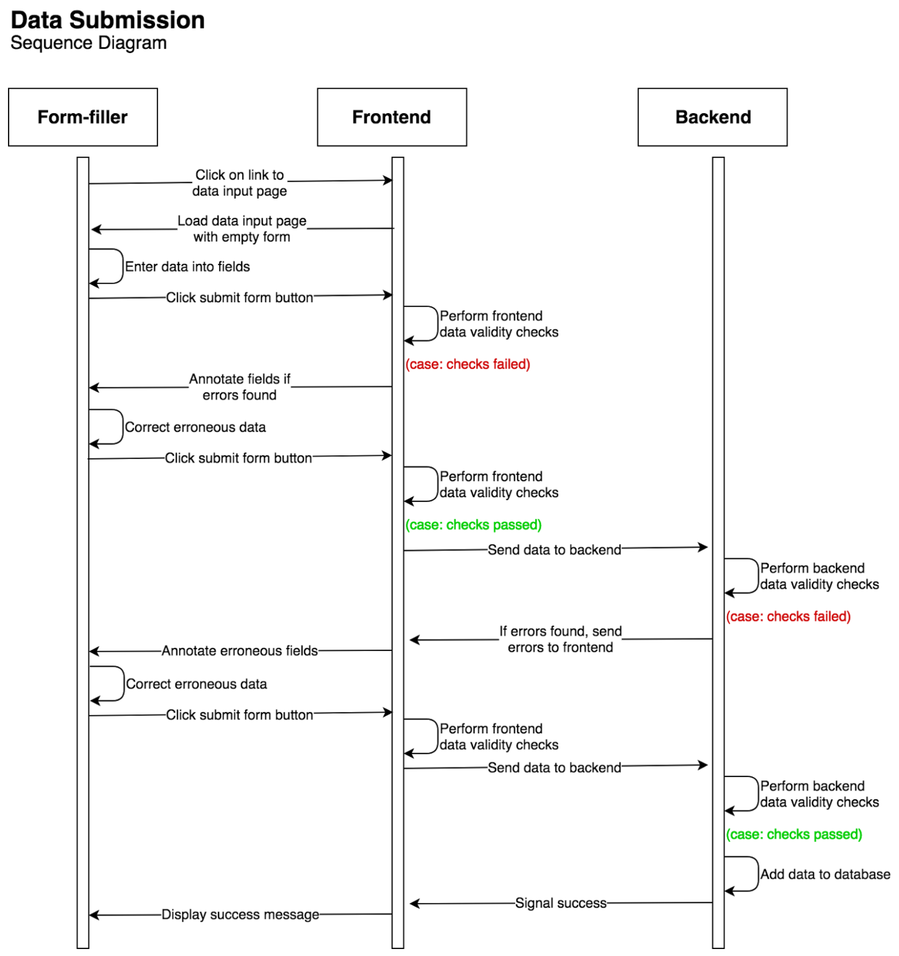


Figure 4: Sequence Diagram for Submitting Data

6.3 CSC and CSU Descriptions

The CaseX CSCI is comprised of three primary CSC's: the Frontend CSC, the Server CSC, and the Database CSC. The CaseX Frontend CSC handles interactions between the end user and the Server CSC. The CaseX Server CSC provides an interface between the Frontend CSC and the Database CSC. The CaseX Database CSC stores all of CaseX's data and organizes it in a useful manner.

6.3.1 Class Descriptions

The responsibilities and functionalities of the CaseX CSC's are described in detail below.

6.3.1.1 Frontend CSC

The CaseX Frontend CSC is a critical component of the CaseX CSCI because of the target end user. The LAPD Homicide Library Unit's detectives have limited experience with technology and require a tailored user experience to maximize their productivity. The Frontend CSC accomplishes this by dynamically hiding or showing functionalities when relevant, visually organizing information using principles of visual design, and providing feedback to the end user in the form of colors and messages.

6.3.1.2 Server CSC

The Server CSC is the main form of communication between the Frontend and Database. Queries are submitted through, validated, and processed by the Server, which responds with the appropriate requested data (granted validation checks pass and the data exists in the Database). The Server is also in charge of validating user privileges upon certain requests (e.g. if a request was made to delete a user, the server checks to make sure the request is generated by an administrator).

6.3.1.3 Database CSC

The Database CSC consists of the MongoDB database, and WiredTiger storage engine. MongoDB is an open-source, document database designed for ease of development and scaling. CaseX uses v3.4.9 of MongoDB. The WiredTiger storage engine is the default storage engine starting in MongoDB 3.2. MongoDB was chosen as the underlying database technology for CaseX because of it high performance, scalability (both vertical and horizontal), and expressive query language. The Database CSC contains MongoDB collections which persist information about cases, suspects, users, and victims, which are stored as documents.

6.3.2 Detailed Interface Descriptions

The following subsections describe how the CaseX CSC's interact in order to maximize the functionality of CaseX.

6.3.2.1 Frontend-Server Interface

The Frontend CSC interfaces with the Server CSC at several points throughout each possible action.

When the end user submits a case, victim, or suspect on the Input page, the Frontend CSC submits the inputted collection of data to the Server CSC. After checking for issues with the Database CSC, the Server CSC sends an error or success message back to the Frontend CSC for visual display.

The Frontend CSC interfaces with the Server CSC in the same manner when the end user creates a new CaseX account on the Administrator page. The Frontend CSC submits the inputted data to the Server CSC, which sends an error or success message back to the Frontend CSC for visual display.

6.3.2.2 Server-Database Interface

The Server CSC interfaces with the Database CSC through a MongoDB connection facilitated by the Mongoose library. For development, the MongoDB database runs locally on the default port (27017), and uses the default storage engine, WiredTiger. The Mongoose library facilitates object modeling, validation, and business logic for MongoDB. The Server CSC includes Mongoose Models, for user, case, suspect, and victim. The Server CSC includes REST API controllers which provide a structured interface to the Database CSC to create, read, update, or delete (CRUD) entries in the user, case, suspect, and victim database. The REST API controllers will filter and forward any errors returned from the Database CS to the Frontend CSC, if there are any.

6.3.3 Detailed Data Structure Descriptions

The underlying data structure used for the Frontend-Server and the Server-Database interfaces is JSON (JavaScript Object Notation). JSON is a syntax for storing and exchanging data that is lightweight and easy for humans to read and write. As a text format that is language independent, data stored as JSON can be easily be exchanged between the server, client, and database. JavaScript objects can easily be converted into JSON, and vice versa.

JSON data is written as key/value pairs. JSON keys must be strings, and JSON values must be a string, number, object (JSON object), array, boolean, or null.

```
var myObj = {
      "DR": 1,
      "Master DR": "1/a",
      "SolvabilityFactor": 1,
      "Suspect": "John Doe,
      "Victim": "Jane Doe",
};
```

A JavaScript object can be converted to JSON via the JSON.stringify() method.

```
var myObj = { "DR":1, "SolvabilityFactor":1, "Suspect":"John Doe };
var myJSON = JSON.stringify(myObj);
```

JSON can be converted to JavaScript objects via the JSON.parse() method.

```
var myJSON = '{ "DR":1, "SolvabilityFactor":3, "Suspect":"Jane Doe" }';
var myObj = JSON.parse(myJSON);
```

6.3.4 Detailed Design Diagrams

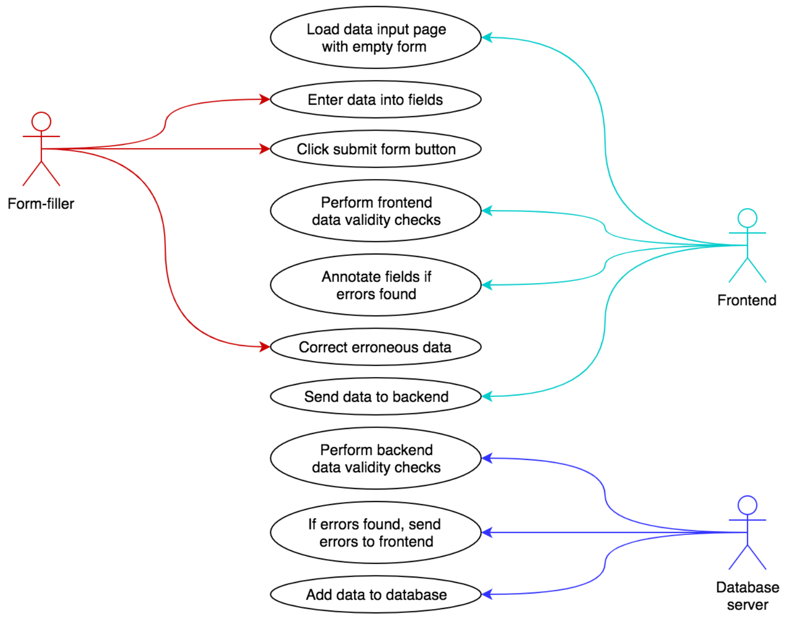

Load data input page with empty form
Enter data into fields
Click submit form button
Perform frontend data validity checks
Annotate fields if errors found
Correct erroneous data
Send data to backend
Perform backend data validity checks
If errors found, send errors to frontend
Add data to database
Form-filler
Frontend
Database server

Figure 6A: Use Case Diagram for Submitting Data

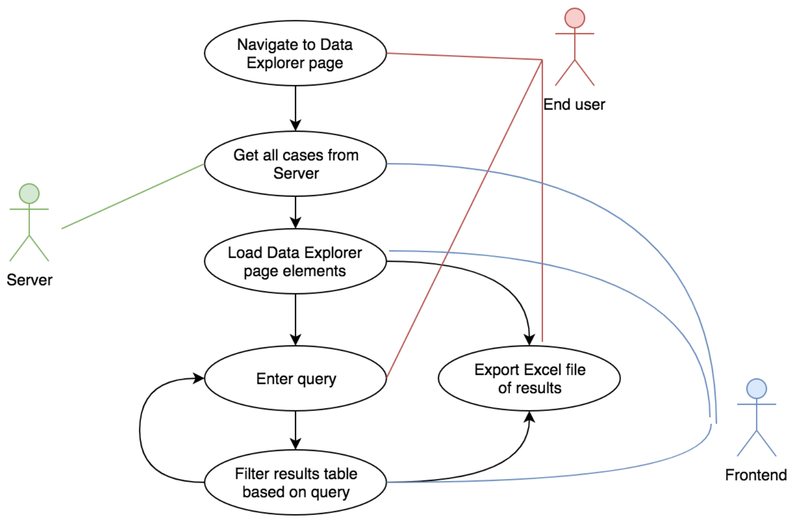

Navigate to Data Explorer page
Get all cases from Server
Load Data Explorer page elements
Enter query
Export Excel file of results
Filter results table based on query
End user
Server
Frontend

Figure 6B: Use Case Diagram for Exploring Data

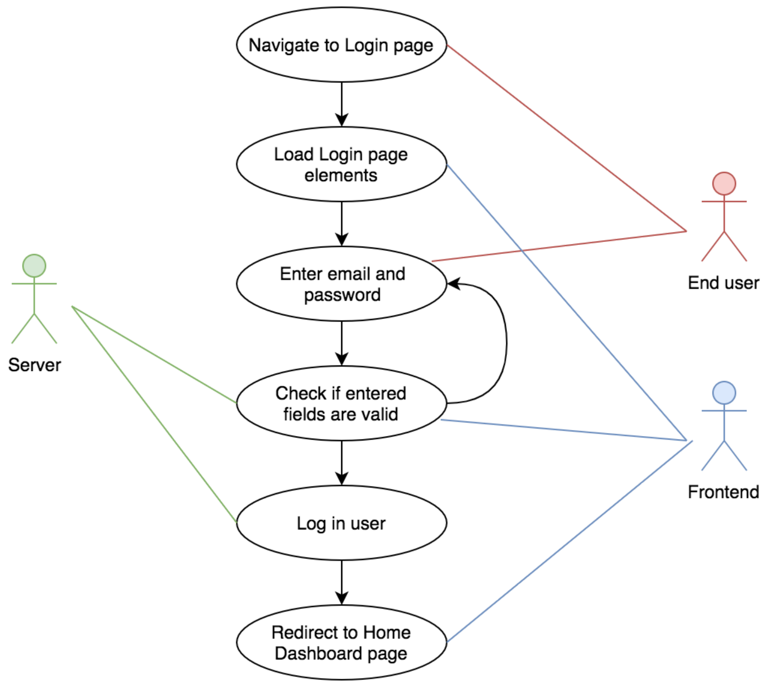


Figure 6C: Use Case Diagram for Logging In

6.4 Database Design and Description

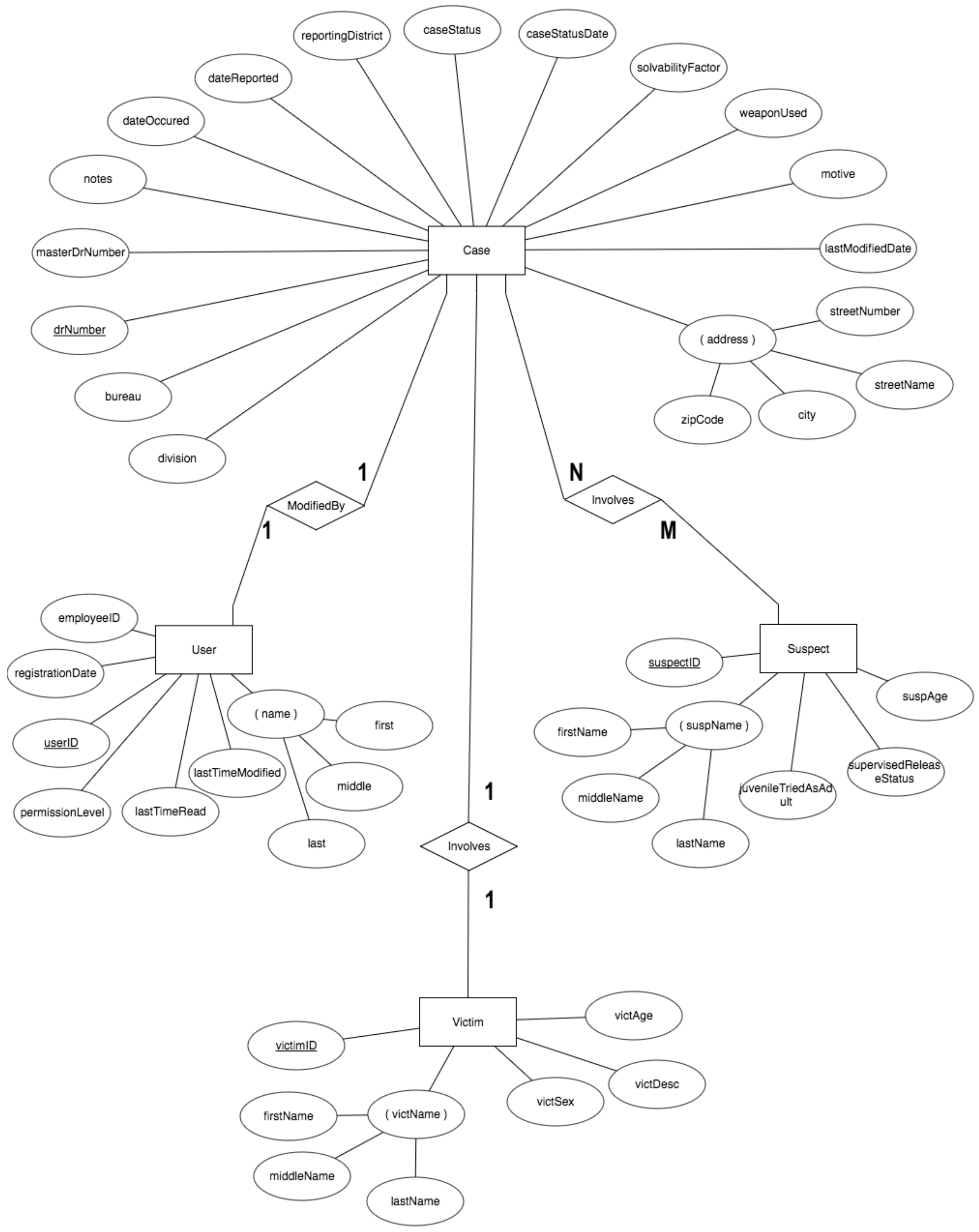
CaseX is an interactive web application and scalable database solution optimized for the management of physical data. The Case Database will be implemented in MongoDB, using Mongoose as a object relational model tool for Node.js. The potential users will be users of the CaseX web application, which will provide a web front-end for this database. The full database for CaseX consists of the **Case Database** and the **User Database.**

The data that will be stored in the Case Database will be the basic data related to each case: the division that is investigating it; the bureau whose jurisdiction it resides in; the DR number for the case, victim(s), and master DR; notes regarding the case; the dates the crime occurred and was reported; the district it was reported in; the status of the case and the date the status was updated; its solvability factor; the address at which it took place; the weapon and/or method used; the motive; and the dates where the case data was last modified and who it was modified by.

The Case Database also contains the Victim, Suspect and Address entities. The Victim entity will be all the data that is related to each victim: the victims' names, the sex and ages of the victims, the description of the victims, and the victims' supervised release statuses. The Suspect entity will include all the data that is related to each suspect: the suspects' names, the sex and ages of the suspects, the description of the suspects, and the suspects’ supervised release statuses. The Address entity comprises of the data in which the incident occurred: the street number, the street name, the city, and the zip code.

The data that will be stored in the User Database will be the data that is related to each user: the user’s ID number, the full name of the user, the level of access that user is permitted, the user’s employee ID number, the dates that the user registered, and the dates that the user last read or modified the data.

6.4.1 Database Design ER Diagram



6.4.2 Database Access

Users of CaseX will interact with the database through the Upload Data, Explore Data, and Generate Reports functionalities. Each of these functionalities are displayed in their respective user interfaces modules. When the user completes an upload data form, the client's browser will run Javascript that makes a POST http request to our server where the database is hosted. When the user submits a query through either the Explore Data interface or Generate Reports interface, the client's browser will run Javascript that makes a GET http request to the server. Additionally, administrators may edit data in the database management section of the administrator console.

6.4.3 Database Security

Security for the CaseX system is achieved through a combination of user authentication and the classification of different types of users, who have varying levels of system privileges. User authentication is implemented using the bcrypt Node.js library module, which hashes user passwords. The three types of user classes are Administrator, Read-Only, and General. Read-Only users cannot input forms or modify user privileges, but have access to view and export data. General users can input forms, modify case data, view and export data, but cannot modify user privileges. Administrators have all of the privileges that General users have, and in addition, are able to modify user privileges (create, update, and delete users from the system).