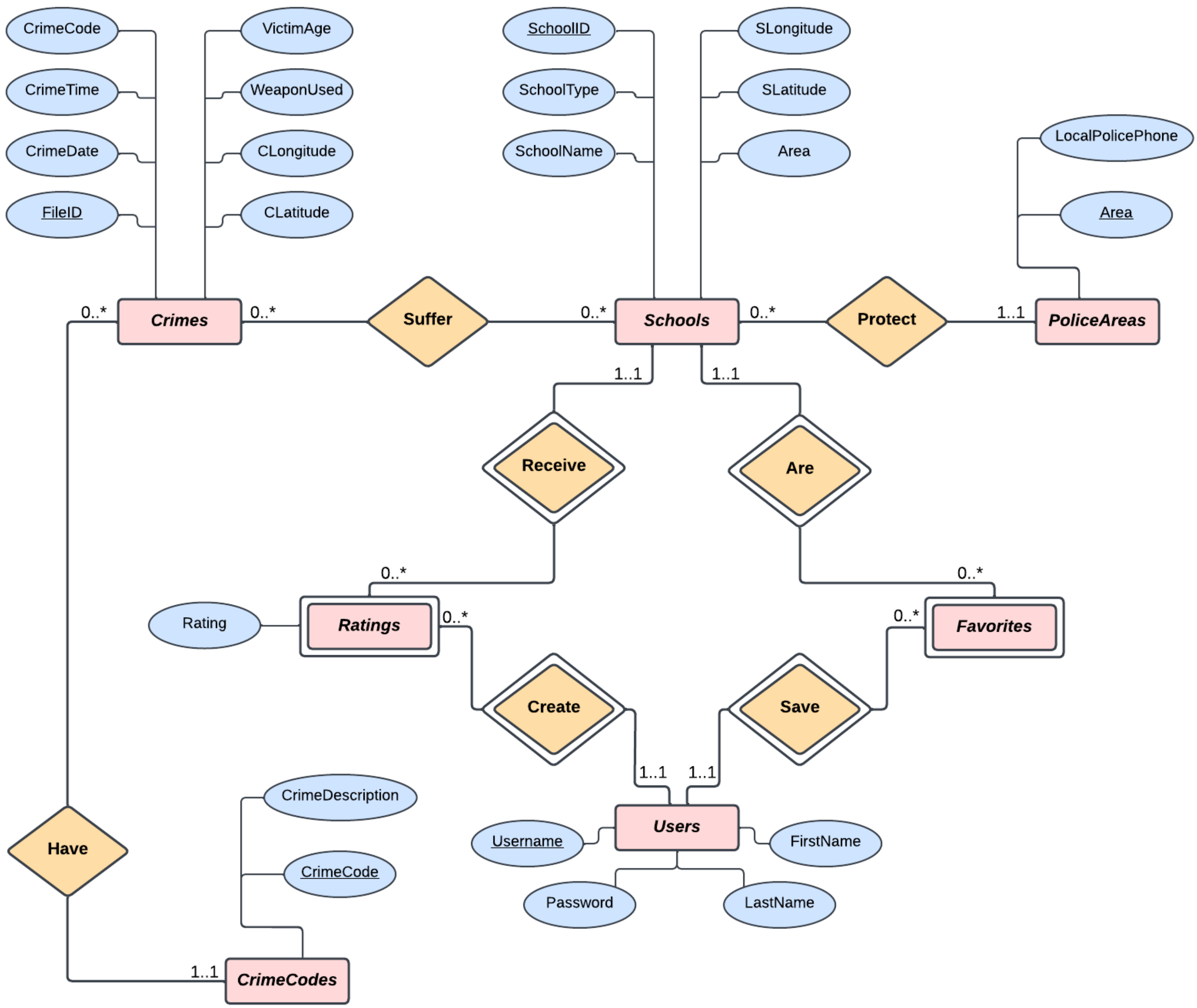
- **ER Diagram**



CrimeCode
CrimeTime
CrimeDate
FileID
VictimAge
WeaponUsed
CLongitude
CLatitude
SchoolID
SchoolType
SchoolName
SLongitude
SLatitude
Area
LocalPolicePhone
Area
0..*
Crimes
0..*
Suffer
0..*
Schools
0..*
Protect
1..1
PoliceAreas
1..1
1..1
Receive
Are
0..*
0..*
Rating
Ratings
0..*
0..*
Favorites
Create
Save
1..1
1..1
CrimeDescription
CrimeCode
Have
Users
Username
FirstName
Password
LastName
1..1
CrimeCodes

- **Assumptions/Descriptions**

Entities:

- *Users*
  - Our Users entity contains information about users that register for our application. We assume that each user has a unique and non-empty username. We also assume that different users can have the same password, first name, and/or last name.
- *Favorites*
  - Our Favorites entity contains information about which users have favorited which schools. We assume that each user can favorite as many different schools as they like (and can also favorite no schools). We also assume that a user cannot favorite the same school twice – this will be enforced at the application level. Additionally, while each favorite corresponds to exactly one user, we assume that multiple users can favorite the same school(s). Finally, we assume that each favorite corresponds to exactly one school.
- *Ratings*
  - Our Ratings entity contains information about users and their ratings for different schools. We assume that each user can write as many reviews as they like (and can also write no reviews). We also assume that each review corresponds to exactly one school. Additionally, we assume that a user cannot write two reviews for the same school – this will be enforced at the application level. Finally, we assume that each review is written by exactly one user.
- *Schools*
  - Our Schools entity contains information about schools throughout LA. We assume that each school has a unique ID. We also assume that different schools might have the same name and we also assume that different schools may have the same latitude and longitude (for instance, if an institution has their middle and high school in the same building).
- *PoliceAreas*
  - Our PoliceAreas entity contains information about different geographical areas in LA and their local police station's contact number. We assume that each geographical area is represented by a unique number. We also assume that two geographical areas may have the same contact number (in the instance where the same police station services two areas).
- *Crimes*
  - Our Crimes entity contains information about crimes committed in LA from 2020 to the present. We assume that each crime is uniquely identified by its associated LAPD file number. We also assume that each crime has exactly one associated crime code.

- *CrimeCodes*
    - Our CrimeCodes entity contains information about the LAPD's crime codes and their associated description. We assume that each crime code is unique and has a non-empty associated and understandable description.

Relationships:

- *Create*
    - Users Create Ratings (One to Many)
    - For our Create relationship, we assume that each user can create anywhere from 0 ratings to ratings for every single school in the database (0..*). We also assume that each individual rating is written by exactly one user – there is no instance where a review can be written by no or more than one user (1..1).
- *Save*
    - Users Save Favorites (One to Many)
    - For our Save relationship, we assume that each user can favorite anywhere from 0 schools to favoriting every single school in the database (0..*). We also assume that each individual favorite is for exactly one user – there is no instance where a favorite is associated with no or more than one user (1..1).
- *Receive*
    - Schools Receive Ratings (One to Many)
    - For our Receive relationship, we assume that each rating corresponds to exactly one school in the database (1..1). We also assume that any individual school can receive anywhere from 0 ratings to ratings from every single user on the application (0..*).
- *Are*
    - Schools Are Favorites (One to Many)
    - For our Are relationship, we assume that each favorite corresponds to exactly one school in the database (1..1). We also assume that any individual school can be marked as a favorite by anywhere from 0 users to every single user on the application (0..*).
- *Suffer*
    - Schools Suffer Crimes (Many to Many)
    - For our Suffer relationship, we assume that each crime can be committed in the vicinity of anywhere from 0 schools to every school in the database (0..*). While, in practice, it might be impossible for one crime to be in the vicinity of every school, there is no way to concretely determine the maximum number of schools that a crime may affect. We also assume that each school can be in the vicinity of anywhere from 0 crimes to every crime in the database (0..*). Once again, while a school being in the vicinity of every crime may be impossible, there is no way to

concretely determine the maximum number of crimes that a school may suffer from.

- *Protect*
    - PoliceAreas Protect Schools (One to Many)
    - For our Protect relationship, we assume that each school has exactly one police precinct protecting it (1..1). We also assume that each police precinct can protect anywhere from no schools to every school in LA (0..*). Following the same logic as above, while it might be impossible that one precinct protects every school, we have no concrete way to determine the maximum number of schools a precinct may protect.
- *Have*
    - Crimes Have CrimeCodes (Many to One)
    - For our Have relationship, we assume that each crime has exactly one crime code associated with it (1..1). We also assume that each type of crime (represented by the crime code) can be committed anywhere from 0 times to being the only crime committed in LA (every record in the Crimes entity has this singular crime code) (0..*).

- **Normalization**

We decided to use 3NF for our database normalization. This was chosen due to the fact that we, as a team, felt we had an overall better understanding of 3NF over BCNF. Our decision was further dictated given the increased ease of achieving lossless decomposition in 3NF versus BCNF. Finally, we also discovered upon completion of the schema that it already adheres to 3NF. This can be shown through the following:

Using the definition of 3NF, we can see that for every non-trivial functional dependency, X → Y, either X is a primary key or every element of Y is part of some candidate key.

To prove this, we first find all of our non-trivial functional dependencies. Given our set of attributes `(SchoolID, SchoolName, SchoolType, Safety, SLongitude, SLatitude, Area, Username, Password, FirstName, LastName, FileID, CrimeDate, CrimeTime, CLongitude, CLatitude, VictimAge, WeaponUsed, CrimeCode, Rating, CrimeDescription, LocalPolicePhone)` and our defined schema, we can observe the following FDs:

```
SchoolID → SchoolName, SchoolType, SLongitude, SLatitude, Area
Username → Password, FirstName, LastName
FileID → CrimeDate, CrimeTime, CLongitude, CLatitude, VictimAge,
WeaponUsed, CrimeCode
```

`CrimeCode → CrimeDescription`
`Area → LocalPolicePhone`

We can confirm that there exist no other functional dependencies in our schema by observing our entities' attributes and their relations to each other. In `Schools`, for every record, it is impossible to identify the value of one attribute based on anything but `SchoolID`. We see the same behavior for our `Users, Crimes, CrimeCodes,` and `PoliceAreas` entities as well – every attribute's value can only be ascertained from the entity's primary key. For our `Ratings` and `Favorites` entities, since these have no primary keys, we observe no dependencies between the attributes. This is why our `Rating` attribute never appears in any FD.

For all of our functional dependencies X → Y listed above, we can see that the X attribute is always a primary key. This confirms to us that our schema adheres to 3NF and no further normalization is required. Additionally, we can confirm that no transitive dependencies exist within any of our entities. We can see this through the following:

Our LEFT attributes include `Username, SchoolID, FileID`
Our MIDDLE attributes include `CrimeCode, Area`

From this information, we know that any transitive dependencies must bridge either `CrimeCode` or `Area`. This is because for a transitive dependency A → C (where A → B and B → C are existing dependencies) to exist, we must have some attribute B that exists in our set of MIDDLE attributes. Since `CrimeCode → CrimeDescription` and `Area → LocalPolicePhone` are the only dependencies that involve our MIDDLE attributes on the left-hand side, our transitive dependency must be `A → CrimeDescription` or `A → LocalPolicePhone`. However, since no such dependencies exist, we can confirm that we have no transitive dependencies. This once again confirms that our schema adheres to 3NF.

- **Relational Schema**

Schools (SchoolID:*INT* **[PK]**, SchoolName:*VARCHAR(100)*, SchoolType:*VARCHAR(100)*, SLongitude:*DECIMAL(7,4)*, SLatitude:*DECIMAL(6,4)*, Area:*INT* **[FK to PoliceAreas.Area]**)

Users (Username:*VARCHAR(20)* **[PK]**, Password:*VARCHAR(30)*, FirstName:*VARCHAR(50)*, LastName:*VARCHAR(60)*)

Crimes (FileID:*INT* **[PK]**, CrimeDate:*VARCHAR(10)*, CrimeTime:*VARCHAR(15)*, CLongitude:*DECIMAL(7,4)*, CLatitude:*DECIMAL(6,4)*, VictimAge:*INT*, WeaponUsed:*VARCHAR(50)*, CrimeCode:*INT* **[FK to CrimeCodes.CrimeCode]**)

Ratings (Username, SchoolID: *VARCHAR(20)* **[FK to Users.Username],** *INT* **[FK to Schools.SchoolID] [PK],** Rating:*DECIMAL(3,1)*)

Favorites (Username, SchoolID: *VARCHAR(20)* **[FK to Users.Username],** *INT* **[FK to Schools.SchoolID] [PK]**)

CrimeCodes (CrimeCode:*INT* **[PK],** CrimeDescription:*VARCHAR(100)*)

PoliceAreas (Area:*INT* **[PK],** LocalPolicePhone:*INT*)

Suffer (FileID, SchoolID: *INT* **[FK to Crimes.FileID],** *INT* **[FK to Schools.SchoolID] [PK]**)