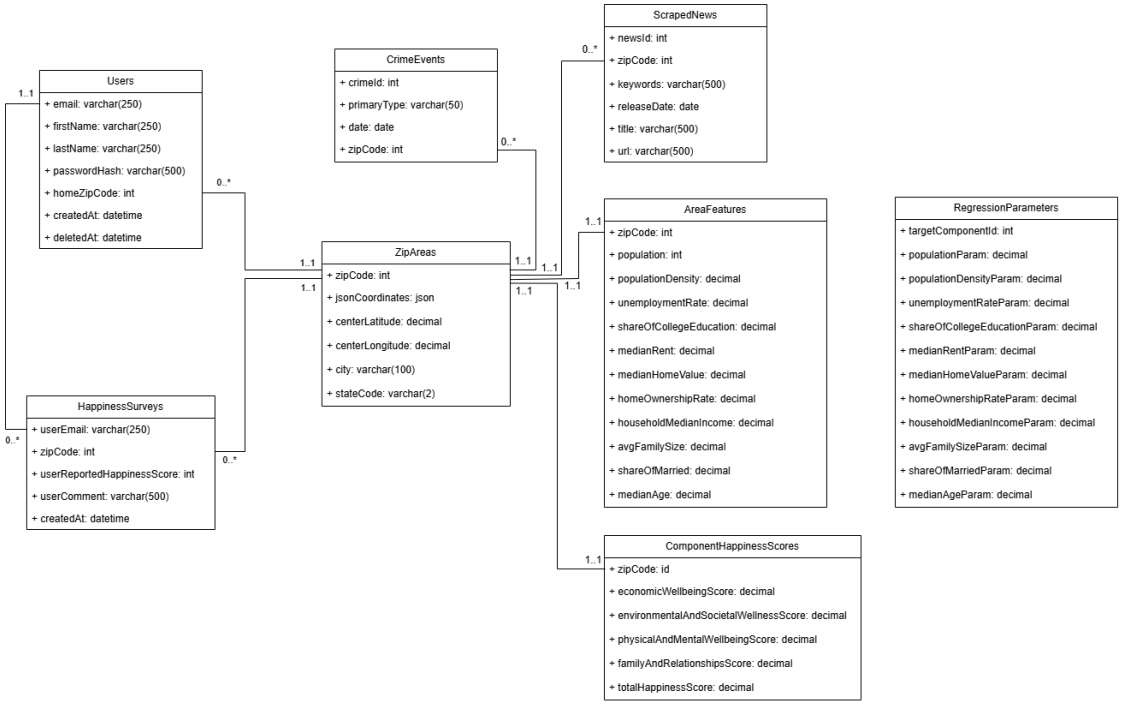
Database Design



Relation Assumptions

Users

1. Users can register with one email only once and, thus, email can be used as a uniquely identifying key.
2. A user must submit one survey when signing up. However, there might be a delay between signing up and submitting a survey, and thus a user might momentarily have zero surveys linked to themselves. If a user moves locations, they must submit another survey and, thus, a user might have multiple HappinessSurveys linked to themselves. This leads to a 0..* relationship.
3. Each user can live in one place and hence are linked to one location defined by a homeZipCode. This links to the ZipAreas entity table, and serves as a way for the user to fetch data about their home location.

HappinessSurveys

1. Each survey submitted by a user is linked to the user by an email.
2. Each survey is about the happiness in one area and, thus, is linked to one ZipAreas entity.
3. A user can provide only one survey per zip code and a composite key of attributes userEmail and zipCode can be used to uniquely identify entries.

ZipAreas

1. Because we only consider the US mainland area, we can use the zipCode as a unique identifier for the entries.
2. The location boundaries of the ZipAreas are stored in json format. Whilst these could be stored in a separate table of coordinates, we have decided to keep these in json format in the table as our web application will need the boundary coordinates in json format for map overlay.
3. Each ZipAreas entry is linked to one AreaFeatures entry and one ComponentHappinessScores entry. This is a one-to-one relationship, because we do not store historical changes in data and only update existing data if necessary.
4. ZipAreas may also have zero-to-many Users, HappinessSurveys, CrimeEvents or ScrapedNews linked to them.

CrimeEvents

1. This table contains data on crime events, including the location, date, and type of crime.
2. The app displays crime data to users based on the provided zip code.
3. Each crime event must be linked to one location and, thus, a one-to-one relationship to the ZipAreas table exists.
4. Each crime entry is given a unique identifier that determines all other attributes.

ScrapedNews

1. We will use a Python package to parse news from the website based on the provided zip code.
2. Once the user searches for the happiness score in a specific location, the app will display news articles relevant to that area.
3. Each scraped news article is given a unique identifier that determines all other attributes.

4. Each ScrapedNews entry must link to one ZipAreas entry via a zipCode. We will derive the zip code from the article contents if available or approximate it based on the article contents.

AreaFeatures

1. This table contains features which we will use to calculate the happiness score based on the zipCode.
2. Each entry is linked to one ZipAreas entry.
3. Since we do not store historical information, the zipCode attribute can serve as a unique identifier for each entry.

ComponentHappinessScores

1. Happiness of a location is estimated with five scores: EconomicWellbeingScore, EnvironmentalAndSocietalWellnessScore, PhysicalAndMentalWellbeingScore, familyAndRelationshipsScore, totalHappinessScore.
2. This table allows users to understand each aspect of the happiness score as well as the overall total happiness score.
3. Each entry in the ComponentHappinessScores table is linked to one ZipAreas entry, as one location can have just one happiness score.

RegressionParameters

1. This table contains the parameter values based on which the individual component scores are calculated. A linear equation representing the product-sum of values in AreaFeatures and RegressionParameters will approximate the component happiness Scores. The parameters in this table are defined by a ML model trained behind-the-scenes on external data.
2. The table will have only 5 entries, identified by the targetComponentId. Each entry represents the parameters of the linear equation for a certain component happiness score.
3. The targetComponentId is an integer value between 0 and 4 (inclusive).

3NF Normalization

For our schema to adhere to 3NF, each relation in the database should have only functional dependencies where the LHS of the equation is a superkey, or the RHS is part of a key. We have already considered this in our database design by separating functional dependencies in different relations. For example, to avoid transitive dependencies we have separated AreaFeatures and ComponentHappinessScores from ZipAreas, and in each table there is one superkey that defines all attributes. We can prove that our database is in 3NF by observing the functional dependencies of each relation.

Users

In the Users table, the email attribute defines all the other attributes and is the only functional dependency.

HappinessSurveys

In the HappinessSurveys table, the component key (userEmail, zipCode) defines all the other attributes and is the only functional dependency. Neither attribute part of the composite key can alone define the rest of the attributes.

ZipAreas

In the ZipAreas table, the zipCode attribute defines all the other attributes and is the only functional dependency.

CrimeEvents

In the CrimeEvents table, the crimeId attribute defines all the other attributes and is the only functional dependency.

ScrapedNews

In the ScrapedNews table, the newsId attribute defines all the other attributes and is the only functional dependency.

AreaFeatures

In the AreaFeatures table, the zipCode attribute defines all the other attributes and is the only functional dependency.

ComponentHappinessScores

In the ComponentHappinessScores table, the zipCode attribute defines all the other attributes and is the only functional dependency.

RegressionParameters

In the RegressionParameters table, the targetComponentId attribute defines all the other attributes and is the only functional dependency.

Relational Schema

```
Users(
     email:varchar(250) [PK],
     firstName: varchar(250),
     lastName: varchar(250),
     passwordHash: binary(512),
     homeZipCode: int,
     createdAt: datetime,
     deletedAt: datetime
)

ZipAreas(
     zipCode: int [PK],
     jsonCoordinates: JSON,
     centerLatitude: decimal,
     centerLongitude: decimal,
     city: varchar(100),
     stateCode: varchar(2)
)

CrimeEvents(
```

```
    crimeId: int [PK],
    primaryType: varchar(50),
    Date: date,
    zipCode: int [FK to ZipAreas.zipCode]
)

ScrapedNews(
    newsId: int [PK],
    zipCode: int [FK to ZipAreas.zipCode],
    keywords: varchar(500),
    releaseDate: date,
    title: varchar(500),
    url: varchar(500)
)

HappinessSurveys(
    userEmail: varchar(250) [PK, FK to Users.email],
    zipCode: int [PK, FK to ZipAreas.zipCode],
    userReportedHappinessScore: int,
    userComment: varchar(500),
    createdAt: datetime
)

AreaFeatures(
    zipCode: int [PK, FK to ZipAreas.zipCode],
    population: int,
    populationDensity: decimal,
    unemploymentRate: decimal,
    shareOfCollegeEducation: decimal,
    medianRent: decimal,
    medianHomeValue: decimal,
    homeOwnershipRate: decimal,
    householdMedianIncome: decimal,
    avgFamilySize: decimal,
    shareOfMarried: decimal,
    medianAge: decimal
)
```

```
ComponentHappinessScores(
      zipCode: int [PK, FK to ZipAreas.zipCode],
      economicWellbeingScore: decimal,
      environmentalAndSocietalWellnessScore: decimal,
      physicalAndMentalWellbeingScore: decimal,
      familyAndRelationshipsScore: decimal,
      totalHappinessScore: decimal
)

RegressionParameters(
      targetComponentId: int [PK],
      populationParam: decimal,
      populationDensityParam: decimal,
      unemploymentRateParam: decimal,
      shareOfCollegeEducationParam: decimal,
      medianRentParam: decimal,
      homeOwnershipRateParam: decimal,
      householdMedianIncomeParam: decimal,
      avgFamilySizeParam: decimal,
      shareOfMarriedParam: decimal,
      medianAgeParam: decimal
)
```

Clarifying Comment on Feedback Received from Stage 1

We have identified a data/source that estimates happiness based on four component happiness scores. However, that data source (Happiest States 2024 (datapandas.org)) is limited to only state level data. Our plan is to use zip level features to train a linear regression model to match the state happiness scores, and then using that model to approximate the happiness scores on zip level data. This data preparation will be done locally, and only the final datasets will be uploaded into the database.