Restaurant-Review-API Project Report

The service is developed using Node.js Express.js and MySQL.

**Code Walkthrough and Directory Structure:**

The project root directory contains the following folders

**Config:** used for storing the environment configuration files.

**Lib:** Contains following sub directories:

**Routes:** Different API Routes for Users, Restaurants, Rating and service status.

**Utils:** Contains reusable helper functions for input validation, error handling and some other reusable modularized code.

**Setup-files**: Contains supporting documentation and files required to setup sql-schema and postman schemas.

**Test:** Unit and Integration tests subfolders for the application.

Due to shortage of time, I was only able to create one unit test file which demonstrates all the key unit testing concepts like stubbing, 100% branch coverage, 100% code coverage and testing of function in isolation by mocking out all the dependency functions.

**Database Design**

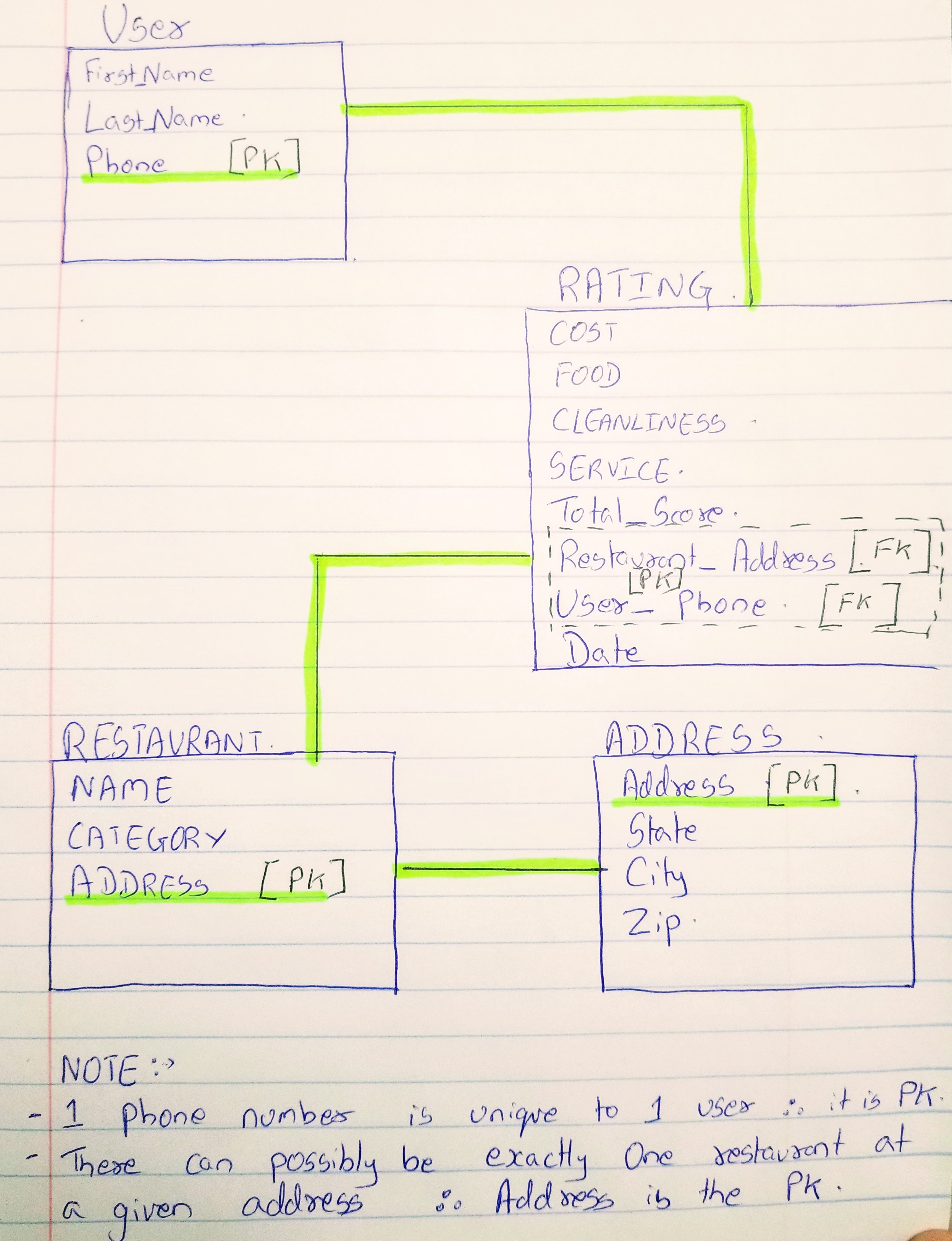
I am using 4 tables here: Users, Restaurants, Ratings and Address.

- Phone number is the primary key in the User table. One user can have only one phone number

- Address is primary key in Restaurant and Address tables. There can be only one possible unique restaurant at a given address.

- In the RATING table, Restaurant address and User Phone are foreign keys. And together they form the foreign key in the Ratings table.

**ER Diagram :**



**Identifying Relationships:**

USER:

User 🡪 Rating; One to Many. One user can submit many ratings.

User 🡪 Address; Null

User 🡪 Restaurant; Null

RESTAURANT:

Restaurant 🡪 User; Null

Restaurant 🡪 Address; One to Many. One restaurant can have many addresses.

Restaurant 🡪 Rating; One to Many. One Restaurant can have many ratings.

ADDRESS:

Address 🡪 User; Null

Address 🡪 Restaurant; Many to 1; Many address can belong to one Restaurant type.

Address 🡪 Rating; One to Many; One Address can have many ratings.

RATING:

Rating 🡪 User; Many to One; Many ratings can be submitted by one single user

Rating 🡪 Restaurant; Many to one; There can be many ratings for each restaurant.

Rating 🡪 Address; Null