David Antaki
Siddharth Saraf
CS 3200

Database Design - Final Report

1. **README (How to setup and run application)**

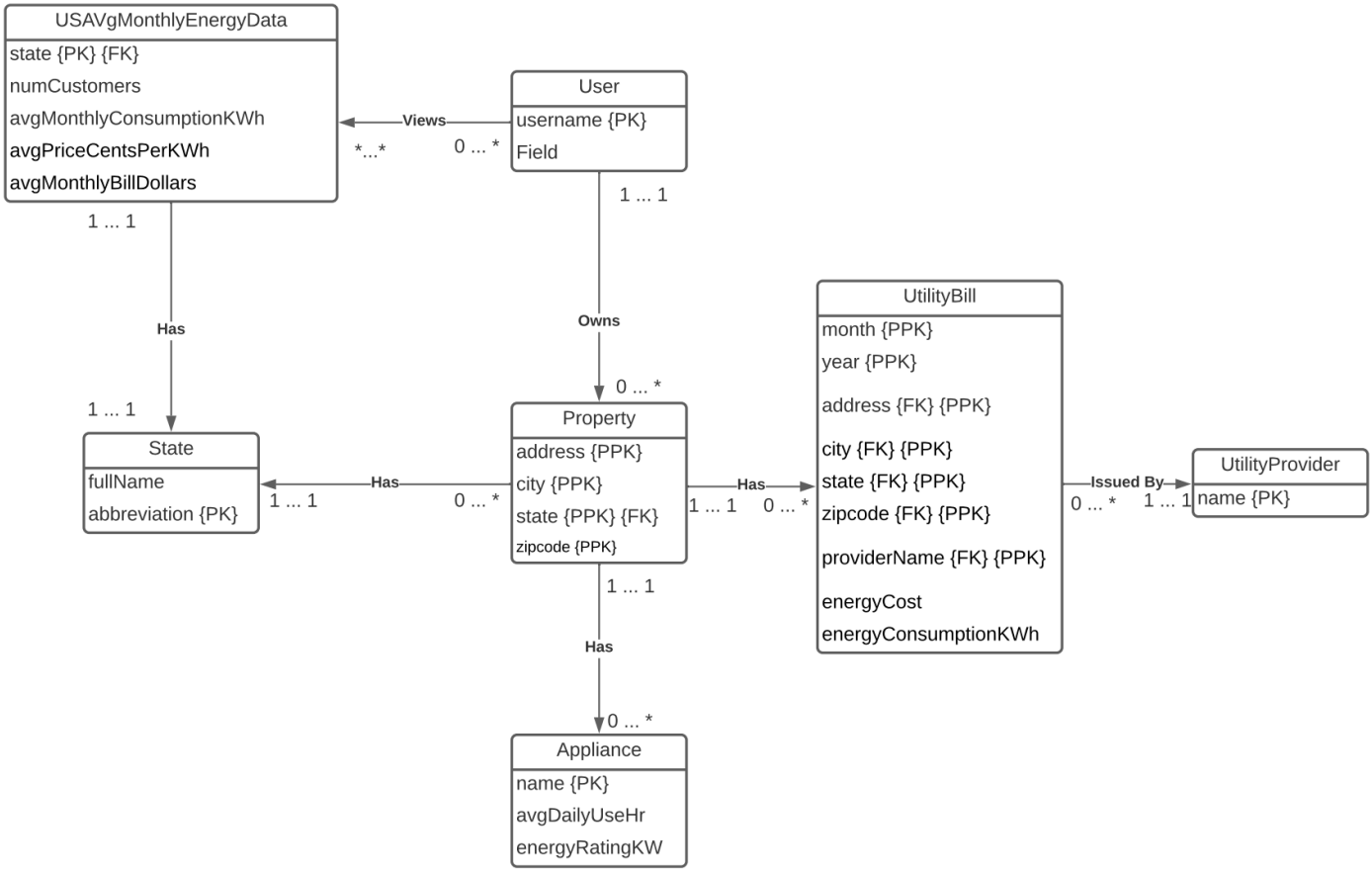
Setup and Running the Application
Setting up the MySQL database
* Install the latest version of MySQL workbench: https://dev.mysql.com/downloads/workbench/
* Create a server instance and connect to it through MySQL.
* In MySQL workbench, import the database dump filed name energy_app_database_dump.sql by going to Server->Data Import
* Select "Import from Self-Contained File" and then select energy_app_database_dump.sql dump file.
* Click start import.
* This will import the database schema, some example tuples, and will add a user with permissions that is used for the connection between this application and the database.
Running the User Application
* Install the latest version of Python3: https://www.python.org/downloads/
* Install the latest version of git to get git bash (or install some terminal if you don't have one already): https://git-scm.com/downloads
* Open a terminal in the root of the submission folder.
* Create a virtual environment (venv): >python -m venv ./venv
* Start the virtual environment (venv): >source venv/Scripts/activate or on Mac run: >source venv/bin/activate
* Install project dependencies from the given requirements.txt: >pip install -r requirements.txt
* Run the user application: >python energy_app.py

2. **Technical Specification**

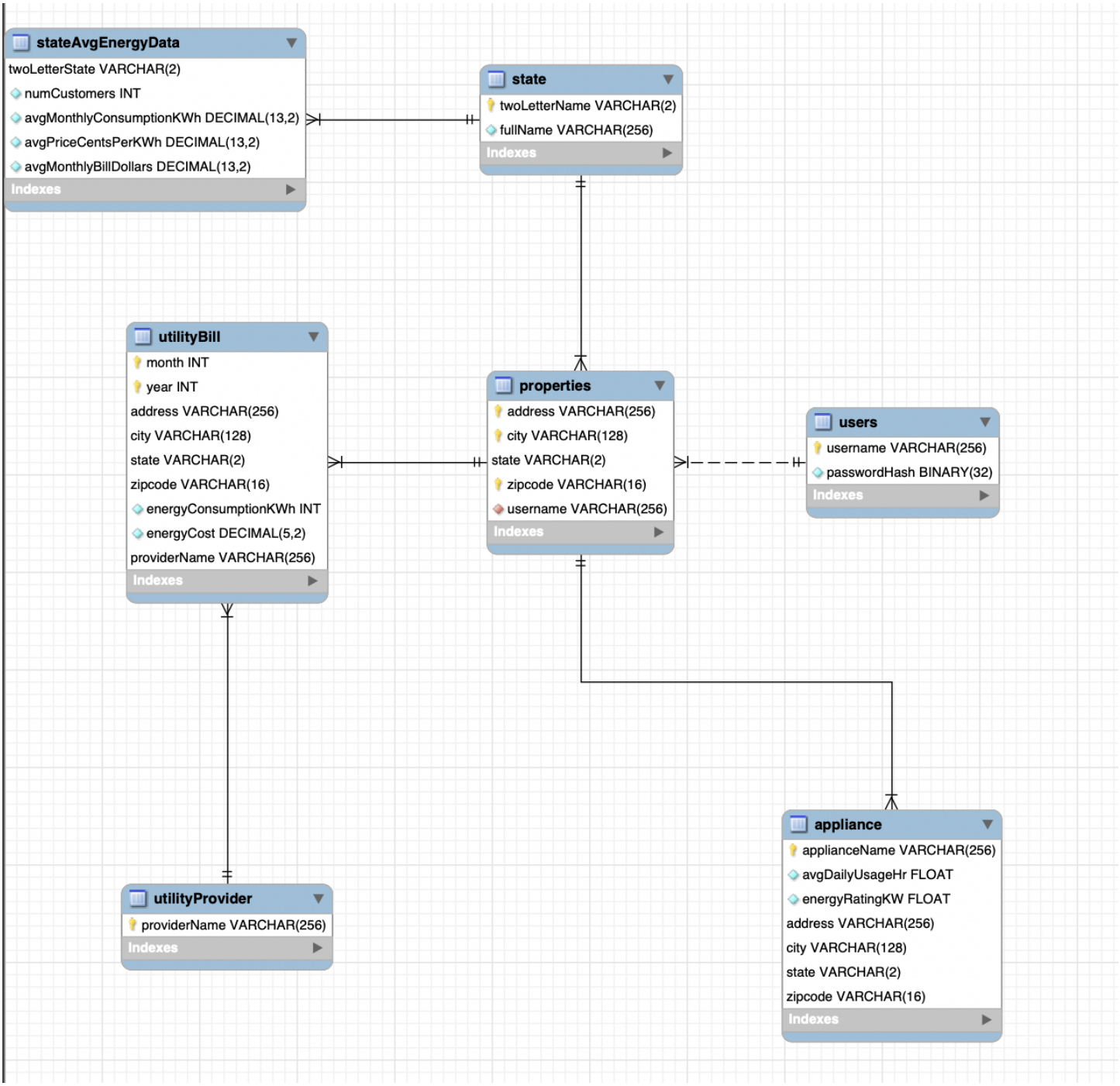
Our data domain is energy usage and is designed for property owners, homeowners, commercial building owners, or landlords who want to track and optimize their buildings energy usage. Our application allows users to input and track their utility bills for multiple properties. Users can track energy usage of appliances. Our application allows users to compare their energy usage to that of national average energy consumption and price data per state.

We used MySQL for the database and a text based Python application for the frontend. We use the pymysql python library for the connection between the SQL database and the user application.

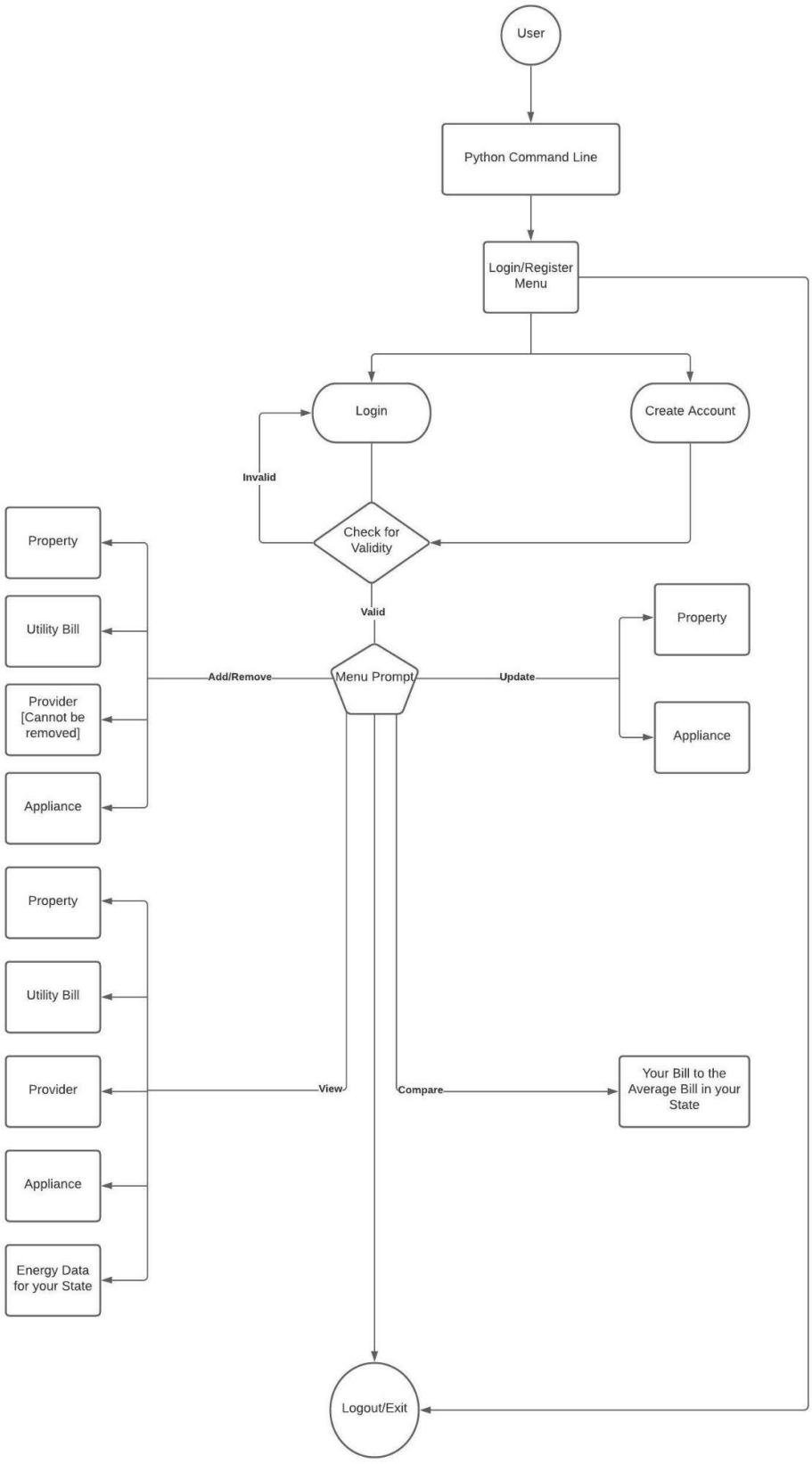
3. **UML Diagram**


USAVgMonthlyEnergyData
state {PK} {FK}
numCustomers
avgMonthlyConsumptionKWh
avgPriceCentsPerKWh
avgMonthlyBillDollars
Views
...
0 ... *
User
username {PK}
Field
1 ... 1
1 ... 1
Has
Owns
1 ... 1
0 ... *
State
fullName
abbreviation {PK}
Has
1 ... 1
0 ... *
Property
address {PPK}
city {PPK}
state {PPK} {FK}
zipcode {PPK}
Has
1 ... 1
0 ... *
UtilityBill
month {PPK}
year {PPK}
address {FK} {PPK}
city {FK} {PPK}
state {FK} {PPK}
zipcode {FK} {PPK}
providerName {FK} {PPK}
energyCost
energyConsumptionKWh
Issued By
0 ... *
1 ... 1
UtilityProvider
name {PK}
1 ... 1
Has
0 ... *
Appliance
name {PK}
avgDailyUseHr
energyRatingKW

4. Logical Database Schema



5. User Flow


User
Python Command Line
Login/Register Menu
Login
Create Account
Invalid
Check for Validity
Valid
Property
Utility Bill
Provider [Cannot be removed]
Appliance
Add/Remove
Menu Prompt
Update
Property
Appliance
Property
Utility Bill
Provider
Appliance
Energy Data for your State
View
Compare
Your Bill to the Average Bill in your State
Logout/Exit

6. **Lessons Learned**
    a. Technical Expertise Gained
        i. Knowledge on how to design and build an application that uses a database for storing data as opposed to storing data in the application itself or in a simple text file.
        ii. Learned how to connect a user facing application (frontend) to a database (backend).
        iii. A deeper understanding of SQL, its connection to Python, and how that can be implemented in real life examples was the main technical expertise gained. The ease with which data can be managed, manipulated, and analyzed was a key lesson that we took while doing the project. The initial front end development with python and back end with SQL may be complex but the results it produces and the ease with which a user can then play around with data was brilliant.
    b. Insights, time management insights, data domain insights etc.
        i. We thought our time was managed well to complete this application.
        ii. About 24 total hours were spent writing the application itself (SQL and Python).
        iii. We essentially divided up the project in phases and then dedicated a few hours to it every few weeks to make sure we were on top of the tasks we needed to do. Making sure we are not biting off more than we can chew was the main concern as implementing an idea that may seem simple is actually difficult and has a lot of technicalities and concerns that we needed to keep in mind. We dealt with such issues as they came and tackled them by either remodeling our project a little or digging deeper for the data that would fit our project's requirements.
    c. Realized or contemplated alternative design / approaches to the project
        i. We did use a couple of alternative approaches when compared to our initial design so that the project was manageable and the application we had performed efficiently and did its job right, rather than an application that did too much but poorly. We got rid of the prediction capability of our project as implementing the basic feature set of tracking utility bills and appliances took longer than expected. Further, the focus of the project was on CRUD operations and on the overall design and implementation of a database. Previously, we were focusing on simply storing data taken from online resources and giving the user the ability to do operations on that data. We slightly changed our focus to give the user the ability to input and update their own data. We kept the ability for the user to compare their data with external "average" data. We added an appliance entity and the ability to do CRUD operations on that data. The ability to predict future bills and give the user more insight into their data and how they could optimize their energy usage is a future improvement.
        ii. Another alternative we considered for the database design was how we purposely made a utilityBill belong to a property and then a utilityProvider belong to a utilityBill. We believed this was the simplest approach. However, it also makes sense that a utilityProvider provides power to a property (a relationship we didn't directly represent) and a utilityProvider also issues bills to a property (a relationship we did represent). We could have made a 3-way relation in a separate table which would represent the relationship between a property, utilityBill, and a utilityProvider in this manner.

7. **Future Work**
    a. Planned uses of the database
        i. The intended use of the database is for a homeowner or a commercial property manager to track their properties' energy usages and costs to give insights into how they could optimize their properties' energy consumption. With many users using the application, the data of all users could be used to provide insights to others users in an anonymous manner (e.g. how well one user's energy usage compares to other users who live in the same area or who have a similar property size, etc.)
    b. Potential areas for added functionality

i. We scrapped the prediction model but in the future, the database can certainly be used to predict the consumption and cost of electricity of a household for a particular month based on the data provided.
ii. The frontend code has a lot of duplication. This is a future improvement for the design. But since the focus was on the overall interaction between a database and a user application, code reuse was not a priority.
iii. The user application code was not tested...unfortunately. This is a future improvement that would make future development significantly faster and make the application more robust.
iv. The code is contained in 2 files only: 1 for SQL, and 1 for python code. In the future, or if this application were to extend, the code should be better segmented out. i.e. SQL schema should be in one file, procedures in another.
v. The MVC model would be ideal to use in the future to separate the view from the model. This would also make the code more testable.
vi. The ability to see how much energy you used for a specific month compared to the national average.
vii. Ability to compare what your energy cost would be if you moved to a different state.
viii. Ability to see how much using an appliance costs for a year based on your bills if you have at least 1 year's worth of bills.
ix. More insights into the user’s data based on more real time energy market data and costs to help optimize which energy source they are using at a given time. Such as using stationary storage during peak demand, while using grid power during low demand.
x. Only the most basic information for each entity was included in this application, following the philosophy of keeping it simple and not adding features or information that are not absolutely necessary. We think more data could be added in the future that would be useful to the user.
xi. Ability to delete a user's entire account.
xii. The ability to MOVE utility bills or appliances between properties. This is currently not supported.