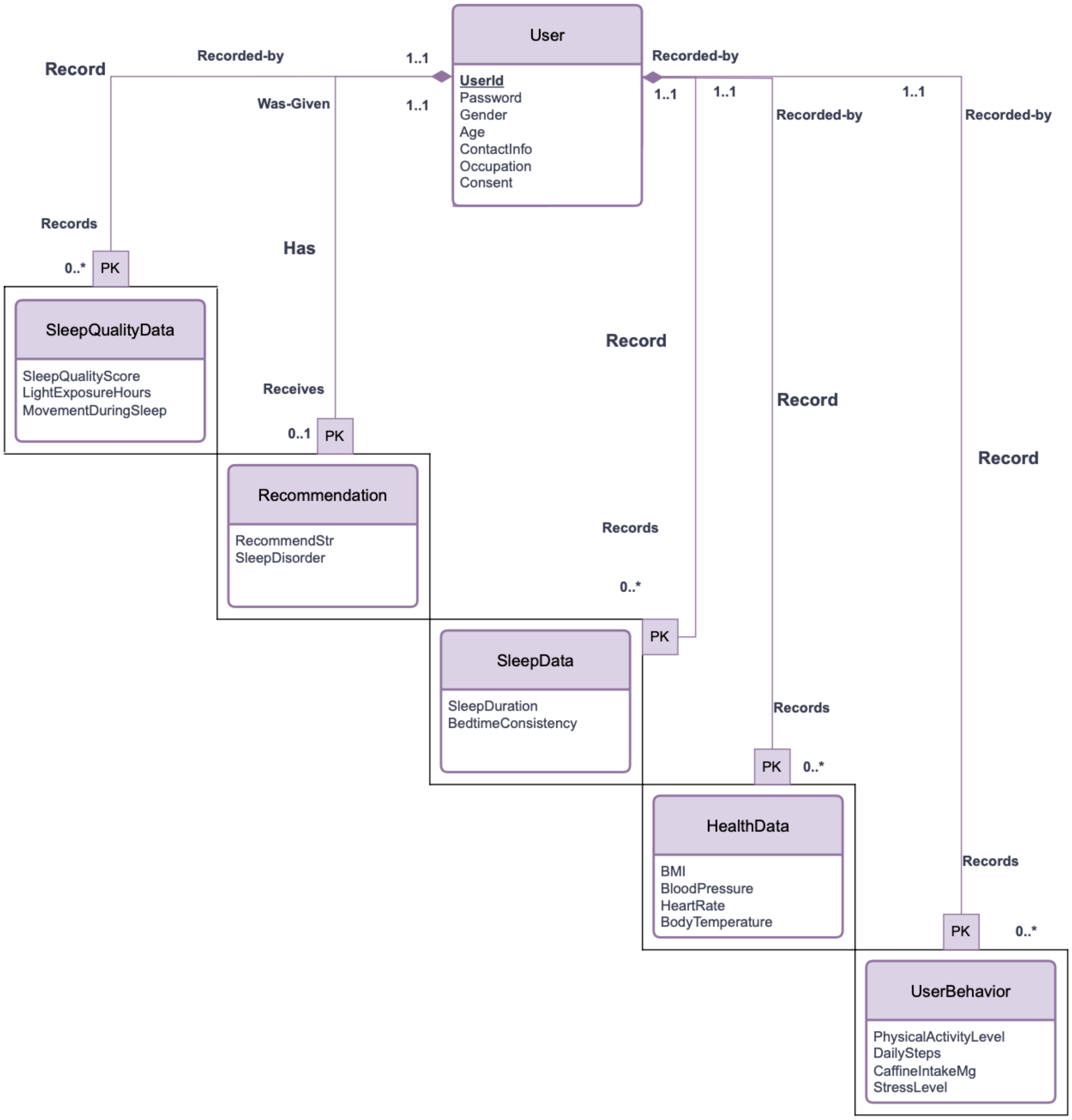
UML diagram & Explanation:



We divide our entities based on the relationship generated by the machine learning model we used to find the relationship between each data category mentioned in the proposal. Users only need to choose one or several of the attributes in an entity other than the User entity, and we will perform different activities based on each entity(though the machine learning algorithm is not perfect, we might need to slightly modify some of the entities accordingly). The user entities contain all the user information we need, and all of those attributes in the user entities need to be entered by the user before any further processing. We will design different checking algorithms for the user entity and other data entities.

The user entity is connected with all the other entities, because all the data of the same user is connected by the UserId, and for the “providing old user personal information” function, we also need to get to the user entity after we find the data of the old user that match, check the consent, and get to the recommendStr in the recommendation, which is the one we end up returning to the new user if the consent of the old user shows “Yes”. A User entity(and a UserId) only link to one set of all the other data entities, but the user can still use our program multiple times, because the UserId in our system will be formed as the true UserId user entered plus the serial number we assigned to each time the user use our program. We do not need to group those data groups that belong to the same user, because the correlation only exists between each UserId and the corresponding data, not between the actual user and their data groups in the dataset-user interaction, and UserId is not involved in the process when we are training our model in the backend.

Normalize the database:

Firstly, we can transform the UML design to the following Relational Schema without domain.

User(UserID [PK], Password, Gender, Age, ContactInfo, Occupation, Consent)

SleepData(UserID [FK to User.UserID PK], SleepDuration, BedtimeConsistency)

HealthData(UserID [FK to User.UserID PK], BMI, BloodPressure, HeartRate, BodyTemperature)

UserBehavior(UserID [FK to User.UserID PK], PhysicalActivityLevel, DailySteps, CaffeineIntakeMg, StressLevel)

SleepQualityData(UserID [FK to User.UserID PK], SleepQualityScore, Ligh- tExposureHours, MovementDuringSleep)

Recommendation(UserID [FK to User.UserID PK], RecommendStr, SleepDis- order)

The we can point out all the FDs, including
UserID→Password,Gender,Age,ContactInfo,Occupation,Consent
UserID→SleepDuration,BedtimeConsistency
UserID→BMI,BloodPressure,HeartRate,BodyTemperature
UserID→PhysicalActivityLevel,DailySteps,CaffeineIntakeMg,StressLevel
UserID→SleepQualityScore,LightExposureHours,MovementDuringSleep

UserID → RecommendStr , SleepDisorder UserID→RecommendStr,SleepDisorder

In all tables, the left side of the FD is the key, also obviously the superkey, so all of them meet the 3NF, there is no need to normalize.

Convert the conceptual database design (ER/UML) to the logical design (relational schema):

User(UserID: INT [PK], Password: VARCHAR(255), Gender: VARCHAR(50), Age: INT, ContactInfo: VARCHAR(255), Occupation: VARCHAR(255), Consent: BOOLEAN)

SleepQualityData(UserID: INT [PK, FK to User.UserID], SleepQualityScore: INT, LightExposureHours: DECIMAL(5,2), MovementDuringSleep: VARCHAR(255))

Recommendation(UserID: INT [PK, FK to User.UserID], RecommendStr: TEXT, SleepDisorder: VARCHAR(255))

SleepData(UserID: INT [PK, FK to User.UserID], SleepDuration: DECIMAL(5,2), BedtimeConsistency: DECIMAL(5,2))

HealthData(UserID: INT [PK, FK to User.UserID], BMI: DECIMAL(5,2), BloodPressure: VARCHAR(50), HeartRate: INT, BodyTemperature: DECIMAL(5,2))

UserBehavior(UserID: INT [PK, FK to User.UserID], PhysicalActivityLevel: VARCHAR(50), DailySteps: INT, CaffeineIntakeMg: DECIMAL(5,2), StressLevel: INT)