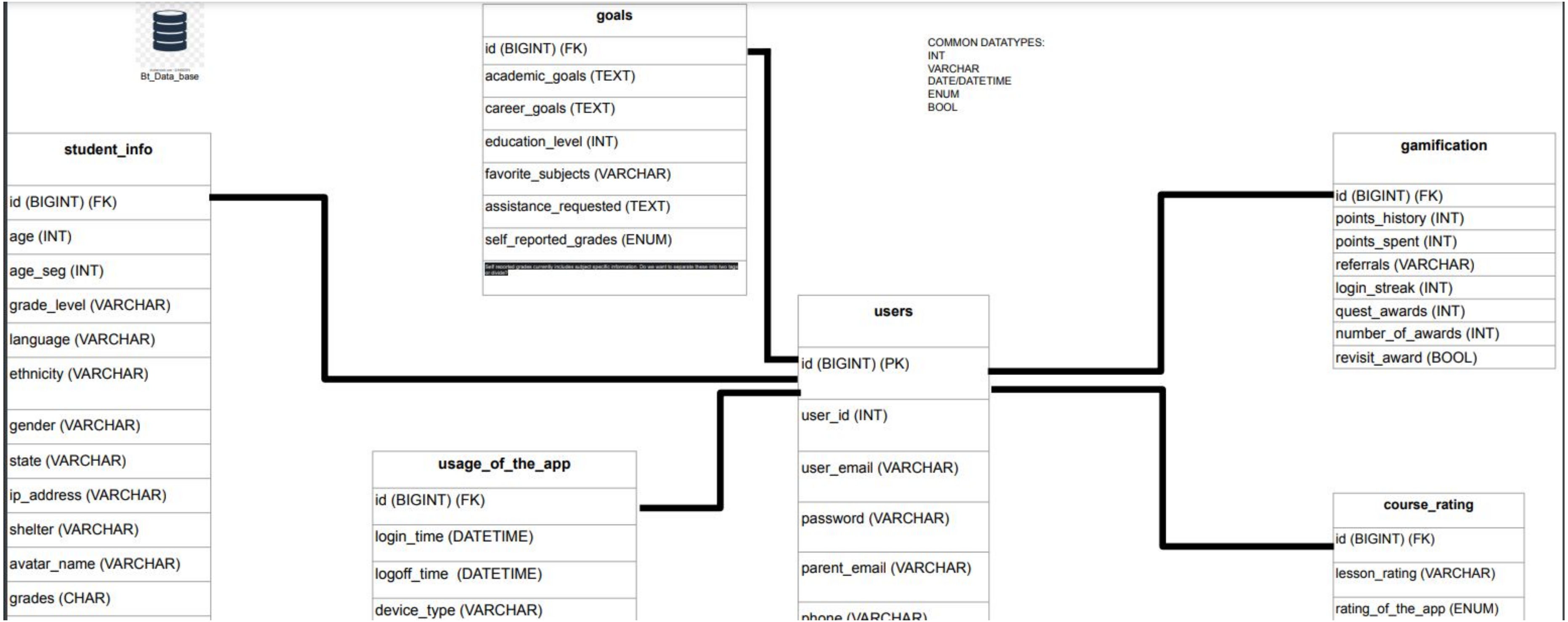
Current Database Schema



COMMON DATATYPES:
INT
VARCHAR
DATE/DATETIME
ENUM
BOOL

goals

- id (BIGINT) (FK)
- academic_goals (TEXT)
- career_goals (TEXT)
- education_level (INT)
- favorite_subjects (VARCHAR)
- assistance_requested (TEXT)
- self_reported_grades (ENUM)

Self reported grades currently includes subject specific information. Do we want to separate these into two tags or divide?

student_info

- id (BIGINT) (FK)
- age (INT)
- age_seg (INT)
- grade_level (VARCHAR)
- language (VARCHAR)
- ethnicity (VARCHAR)
- gender (VARCHAR)
- state (VARCHAR)
- ip_address (VARCHAR)
- shelter (VARCHAR)
- avatar_name (VARCHAR)
- grades (CHAR)

users

- id (BIGINT) (PK)
- user_id (INT)
- user_email (VARCHAR)
- password (VARCHAR)
- parent_email (VARCHAR)
- phone (VARCHAR)

usage_of_the_app

- id (BIGINT) (FK)
- login_time (DATETIME)
- logoff_time (DATETIME)
- device_type (VARCHAR)

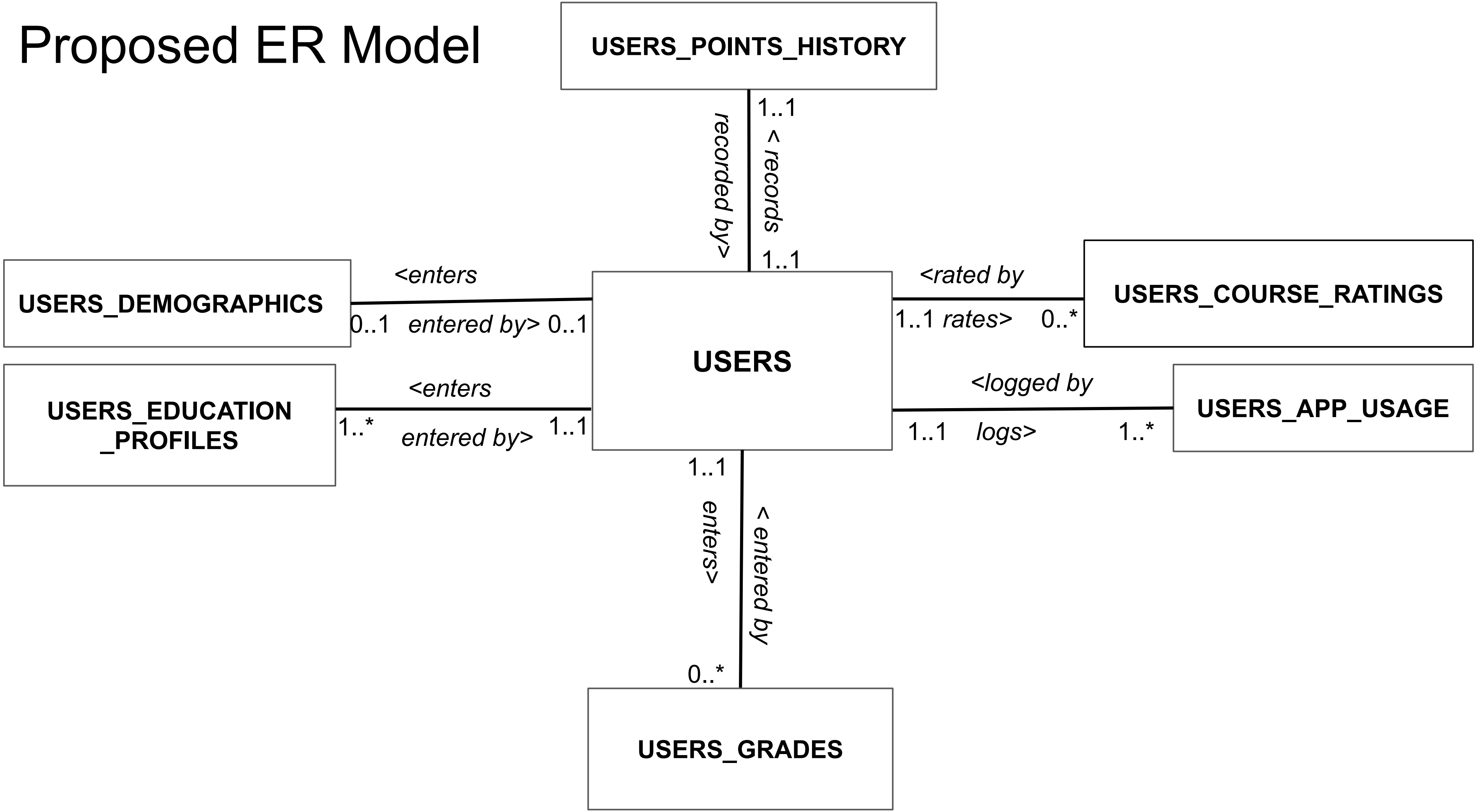
gamification

- id (BIGINT) (FK)
- points_history (INT)
- points_spent (INT)
- referrals (VARCHAR)
- login_streak (INT)
- quest_awards (INT)
- number_of_awards (INT)
- revisit_award (BOOL)

course_rating

- id (BIGINT) (FK)
- lesson_rating (VARCHAR)
- rating_of_the_app (ENUM)

Proposed ER Model



USERS_POINTS_HISTORY

1..1

recorded by>

< records

1..1

USERS_DEMOGRAPHICS

<enters

0..1 entered by> 0..1

USERS

<rated by

USERS_COURSE_RATINGS

1..1 rates> 0..*

USERS_EDUCATION_PROFILES

<enters

1..* entered by> 1..1

<logged by

USERS_APP_USAGE

1..1 logs> 1..*

1..1

enters>

< entered by

0..*

USERS_GRADES

Proposed ER Model: Notes

1. Descriptive entity names
   a. USERS
   b. self_reported_grades → USERS_GRADES
   c. GOALS → USERS_EDUCATION_PROFILES
   d. STUDENT_INFO → USERS_DEMOGRAPHICS
   e. USAGE_OF_THE_APP → USERS_APP_USAGE
   f. GAMIFICATION → USERS_POINTS_HISTORY
   g. COURSE_RATING → USERS_COURSE_RATINGS

2. New entity: **USERS_GRADES**
   a. Rationale: Supports efficient data management and data retrieval. Also allows for better scalability.

Proposed ER Model: Notes, continued

Relationships

a. **USER TO USER_POINTS_HISTORY and USER_POINTS_HISTORY to USER**

   i. USER has a record of their points in USER_POINTS_HISTORY

   ii. Points in USER_POINTS_HISTORY can be recorded by one USER

b. **USER TO USER_COURSE_RATING and USER_COURSE_RATING TO USER**

   i. USER may rate zero to many USER_COURSE_RATING
   ii. Each USER_COURSE_RATING is rated by one USER

c. **USER TO USER_APP_USAGE and USER_APP_USAGE TO USER**

   i. USER may log zero to many USER_APP_USAGE

   ii. Each USER_APP_USAGE is logged by one USER

Proposed ER Model: Notes, continued

Relationships, continued

a. **USER TO USER_GRADES and USER_GRADES TO USER**

   i. USER may have zero to many USER_GRADES instances
   ii. Each USER_GRADES is associated with one USER

b. **USER TO USER_EDUCATION_PROFILE and USER_EDUCATION_PROFILE TO USER**

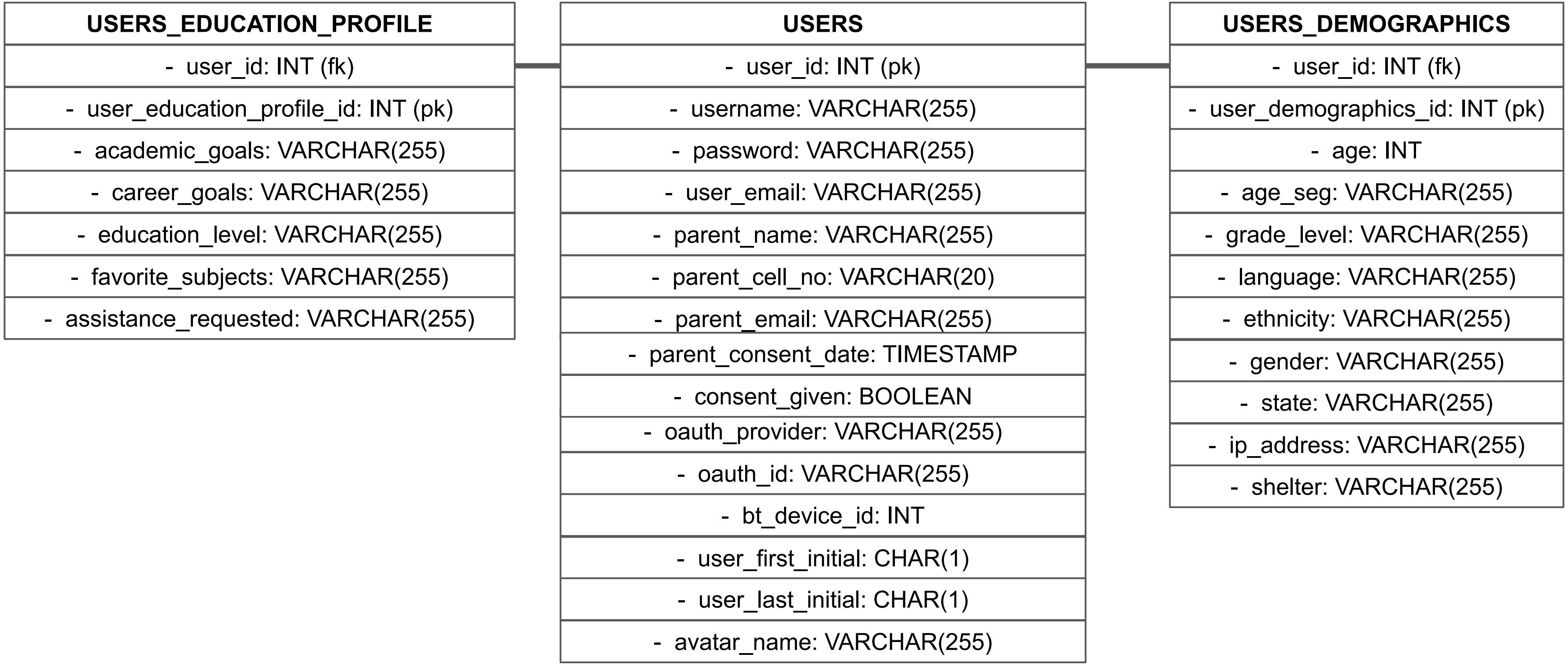
   i. USER enters one or many USER_EDUCATION_PROFILE instances
   ii. Each USER_EDUCATION_PROFILE instance is entered by one USER

c. **USER TO USER_DEMOGRAPHICS and USER_DEMOGRAPHICS TO USER**

   i. USER may enter zero or one USER_DEMOGRAPHICS instance

   ii. Each USER_DEMOGRAPHICS instance is entered by USER

Proposed Relational Model

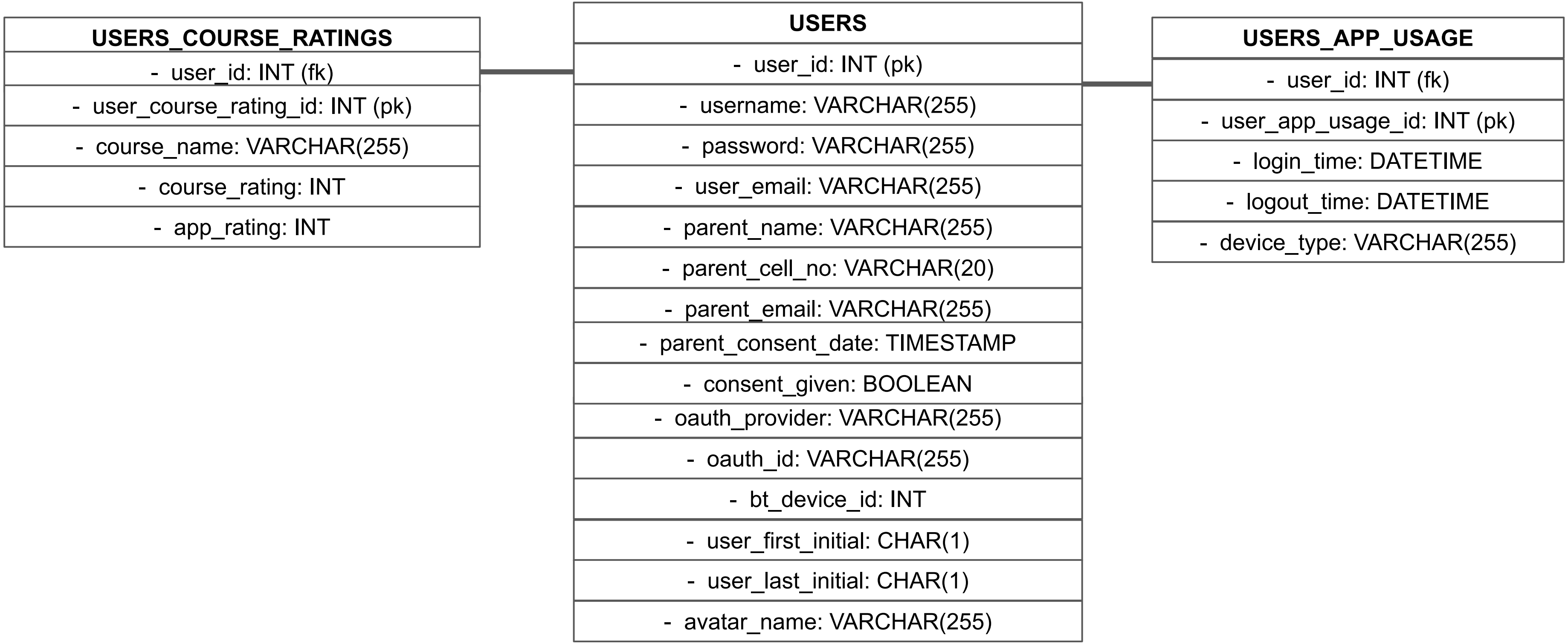


USERS_EDUCATION_PROFILE
- user_id: INT (fk)
- user_education_profile_id: INT (pk)
- academic_goals: VARCHAR(255)
- career_goals: VARCHAR(255)
- education_level: VARCHAR(255)
- favorite_subjects: VARCHAR(255)
- assistance_requested: VARCHAR(255)

USERS
- user_id: INT (pk)
- username: VARCHAR(255)
- password: VARCHAR(255)
- user_email: VARCHAR(255)
- parent_name: VARCHAR(255)
- parent_cell_no: VARCHAR(20)
- parent_email: VARCHAR(255)
- parent_consent_date: TIMESTAMP
- consent_given: BOOLEAN
- oauth_provider: VARCHAR(255)
- oauth_id: VARCHAR(255)
- bt_device_id: INT
- user_first_initial: CHAR(1)
- user_last_initial: CHAR(1)
- avatar_name: VARCHAR(255)

USERS_DEMOGRAPHICS
- user_id: INT (fk)
- user_demographics_id: INT (pk)
- age: INT
- age_seg: VARCHAR(255)
- grade_level: VARCHAR(255)
- language: VARCHAR(255)
- ethnicity: VARCHAR(255)
- gender: VARCHAR(255)
- state: VARCHAR(255)
- ip_address: VARCHAR(255)
- shelter: VARCHAR(255)

Proposed Relational Model (continued)

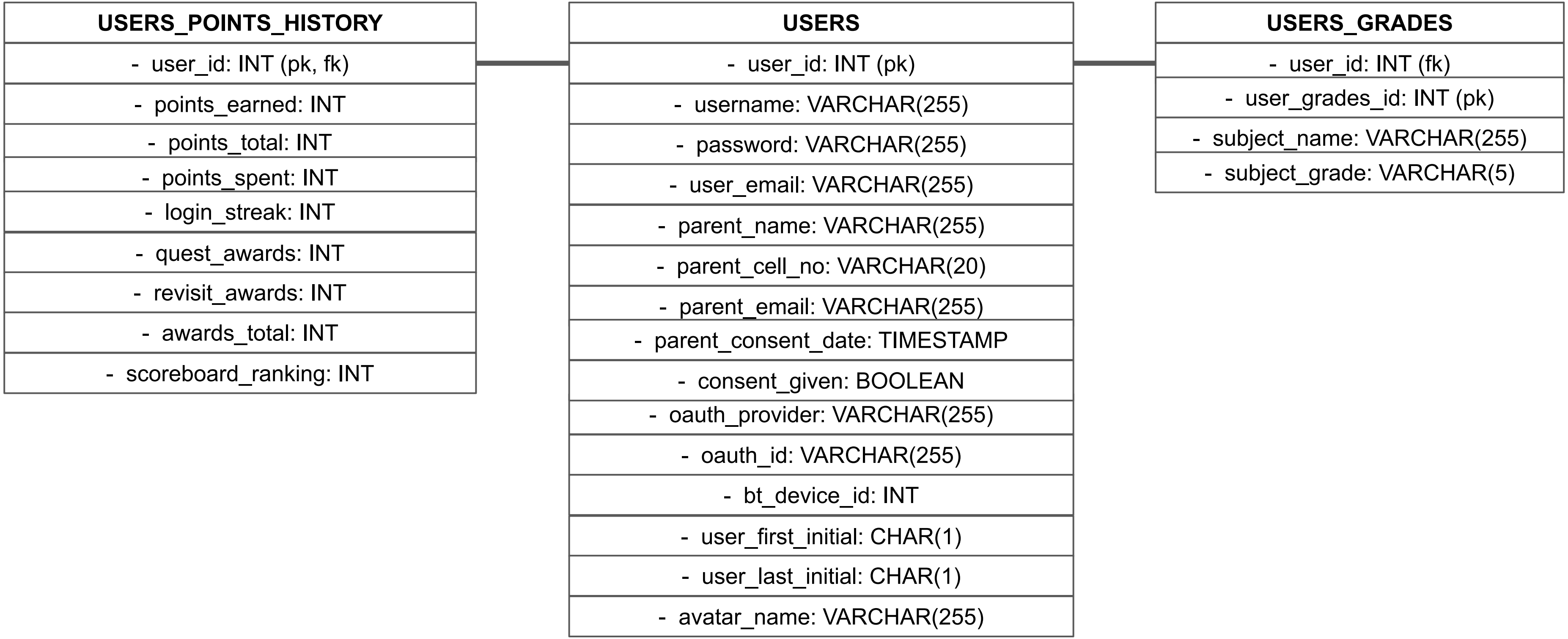


USERS_COURSE_RATINGS
- user_id: INT (fk)
- user_course_rating_id: INT (pk)
- course_name: VARCHAR(255)
- course_rating: INT
- app_rating: INT

USERS
- user_id: INT (pk)
- username: VARCHAR(255)
- password: VARCHAR(255)
- user_email: VARCHAR(255)
- parent_name: VARCHAR(255)
- parent_cell_no: VARCHAR(20)
- parent_email: VARCHAR(255)
- parent_consent_date: TIMESTAMP
- consent_given: BOOLEAN
- oauth_provider: VARCHAR(255)
- oauth_id: VARCHAR(255)
- bt_device_id: INT
- user_first_initial: CHAR(1)
- user_last_initial: CHAR(1)
- avatar_name: VARCHAR(255)

USERS_APP_USAGE
- user_id: INT (fk)
- user_app_usage_id: INT (pk)
- login_time: DATETIME
- logout_time: DATETIME
- device_type: VARCHAR(255)

Proposed Relational Model (continued)



USERS_POINTS_HISTORY
- user_id: INT (pk, fk)
- points_earned: INT
- points_total: INT
- points_spent: INT
- login_streak: INT
- quest_awards: INT
- revisit_awards: INT
- awards_total: INT
- scoreboard_ranking: INT

USERS
- user_id: INT (pk)
- username: VARCHAR(255)
- password: VARCHAR(255)
- user_email: VARCHAR(255)
- parent_name: VARCHAR(255)
- parent_cell_no: VARCHAR(20)
- parent_email: VARCHAR(255)
- parent_consent_date: TIMESTAMP
- consent_given: BOOLEAN
- oauth_provider: VARCHAR(255)
- oauth_id: VARCHAR(255)
- bt_device_id: INT
- user_first_initial: CHAR(1)
- user_last_initial: CHAR(1)
- avatar_name: VARCHAR(255)

USERS_GRADES
- user_id: INT (fk)
- user_grades_id: INT (pk)
- subject_name: VARCHAR(255)
- subject_grade: VARCHAR(5)

User Stories

Basis for CRUD Operations and API Endpoints

User Sign Up (Monica)

- when a new user signs up on the form, then we should create a new user_id in the users database
    - CREATE: POST to /users/ (root)
        - What will be the response to this POST request?
            - HTTP 201 and echo back the user that was created in users table
    - READ: GET /users/count
        - returns 200 and count of users in users table
    - READ: GET/users/list
        - returns 200 and a list of all usernames
    - No UPDATE and No DELETE
    - Error Handling
        - Unique username

USERS
- user_id: INT (pk)
- username: VARCHAR(255)
- password: VARCHAR(255)
- user_email: VARCHAR(255)
- user_first_initial: CHAR(1)
- user_last_initial: CHAR(1)

User Activity: Points (Monica)

- When a user does x activity, we want to add x points to total_points. ~~U~~
- When a user does x activity, we want to add x points to points_earned.~~U~~
- When a user logs out, points_earned will reset to 0. ~~U~~
- When a user logs in on consecutive days, then increment log_in streak. ~~U~~
- When a user wants to see their points history, it will be available on their profile page. ~~R~~
- Activities that will affect points
    - Logging in
    - Watching a video (Clicking on a video link)
    - First time sign up
        - A user_points_history row will be created when a new user is created. ~~C~~
    - Logging out
- Error Handling
    - Ensure points >= 0 (future build – no functionality to spend points yet)

USERS_POINTS_HISTORY
- user_id: INT (pk,fk)
- points_earned: INT
- points_total: INT
- login_streak: INT