Design Document for Immaculate Taste

Group KM_203

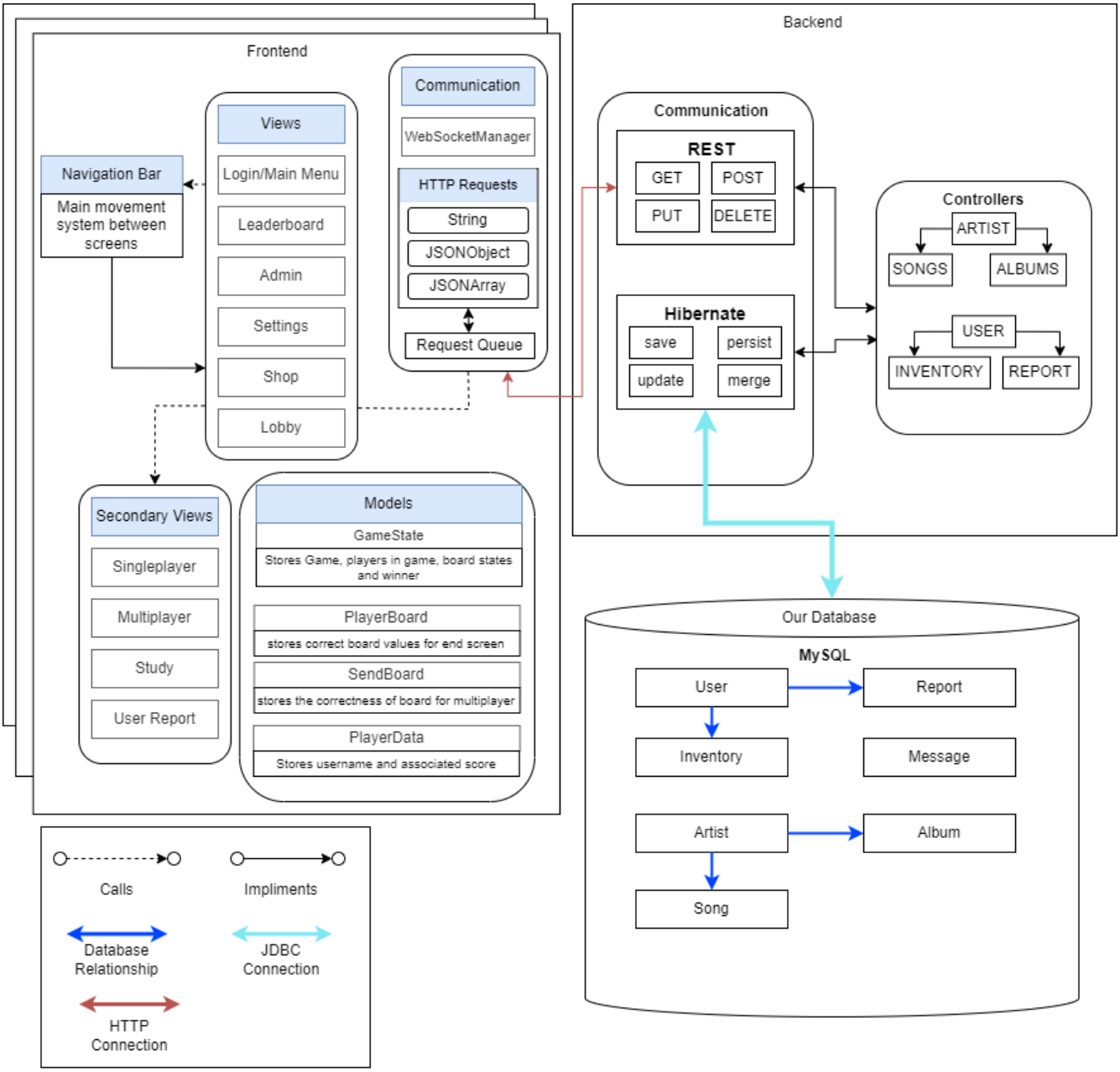
Carter Cutsforth: 25% contribution

Keenan Jacobs: 25% contribution

Conor O'Shea: 25% contribution

Sam Lickteig: 25% contribution

PUT THE BLOCK DIAGRAM PICTURE ON THIS PAGE! (Create the picture using pencil or drawIO)



FrontEnd

BoardState

-BoardState stores the values the player enters in a grid mirroring the current board state. It starts empty, and on every correct value answered in the grid, it appends to the grid at its exact position on the board.

SinglePlayer

-In Singleplayer, the process begins with fetching categories from the backend, which will be used to run the game. The backend sends a list of objects (text, subject, check, and keyword). Text is a string message. Subject can be an artist or a song, check whether a value "contains" or is "with" another, and the keyword is the expected answer. These elements are used in validating answers and setting the boardState object for endgame display. User-entered answers affect the score, and upon completing the board, a call is made to conclude the game, showcasing the board, points, and exiting the game.

Multiplayer

- In multiplayer mode, the same functions as single-player occur. When a player makes a correct move, the multiplayer class generates a copy of the current board (via the sendBoard object) and sends it to the backend with the appropriate message. The other players' UI processes this message, displaying the opposing player's board state. The board also carries information on whether the game has ended, marked as a boolean. If a player receives a message indicating the opposing player has lost, the game ends on their side, showing the loss screen.

BackEnd

Communication

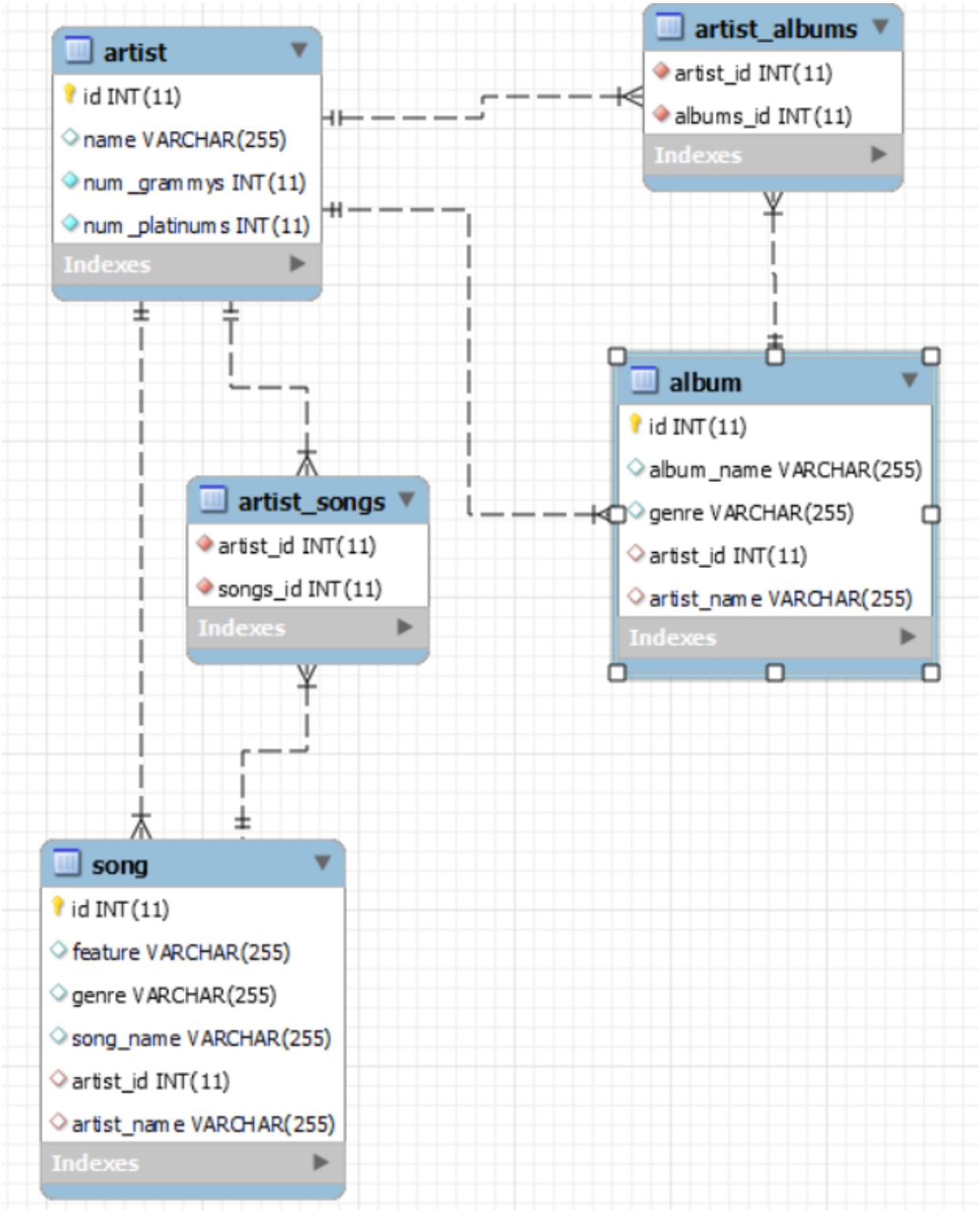
- Post: Used to add the various objects to the database. That includes Songs, Artists, Albums, Users, and Reports.
- Get: Get requests retrieve the data fitting the specifications, for example: finding an Artist in the database by their name. Or getting the list of all Songs an Artist has.
- Put: Used to add things to the database, like creating a new user or updating their high score after they complete a game.
- Delete: Used to delete specified information from the database.

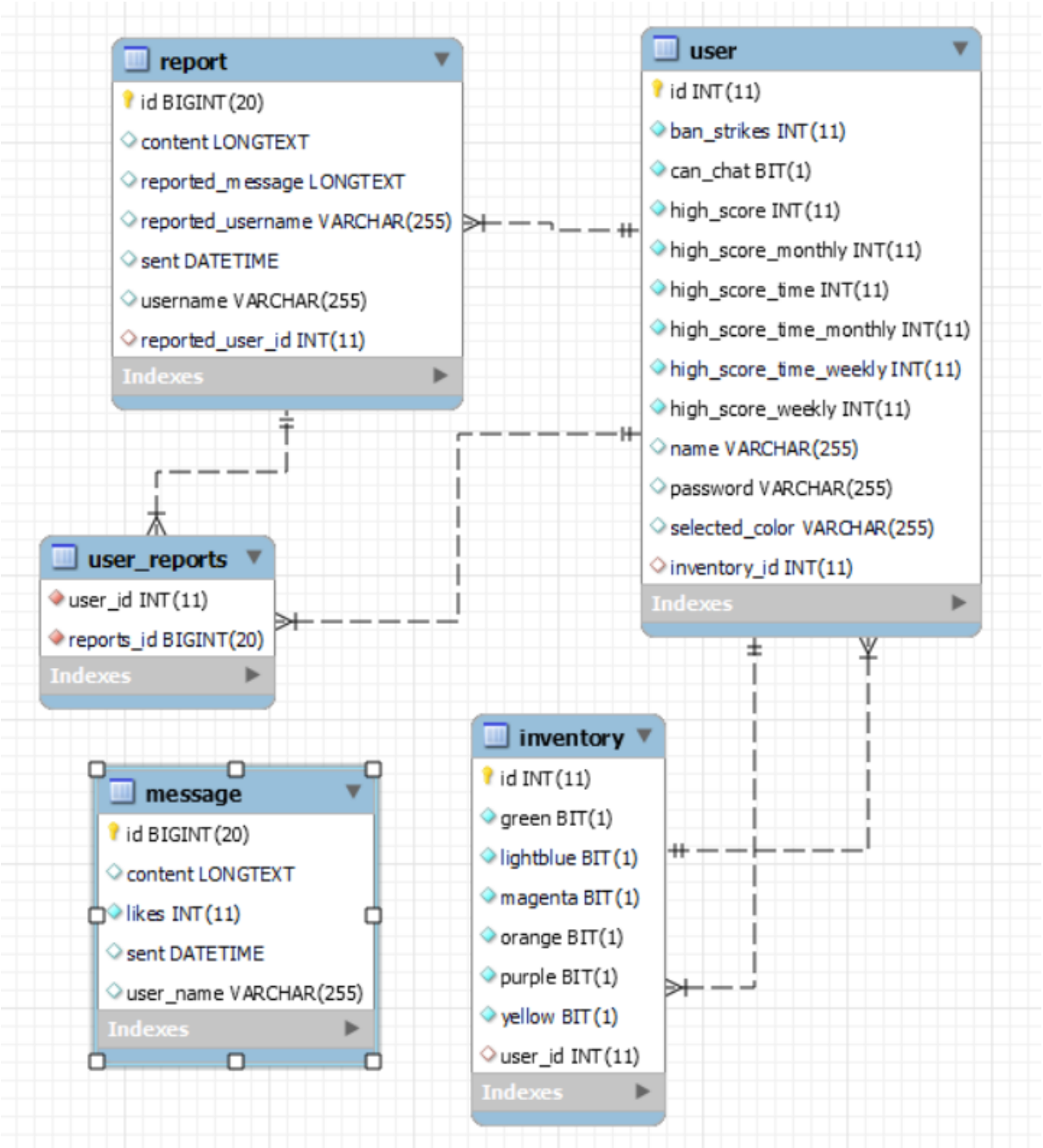
Controllers

- Artist: Contains mappings to allow for creation of Artists, which have a list of their Songs with a one to many relationship (one Artist to many Songs), Albums with another one to many relationship (one Artist to many Albums), grammys, and platinums. Also mappings to search the database for an artist by name or id, and check their song list for specific songs.
- Song: Contains mappings to allow for creation of Songs, which have their Artist with a many to one relationship (many Songs to one Artist), genre, and any features on the song. The mappings can search the database for a song and check which artist made the song, and can check which Artists feature on a given song.
- Album: Contains mappings to allow for creation of Albums, which have their Artist with a many to one relationship (many Albums to one Artist) and the genre. The mappings can search the database for an album and check which artist made it.
- User: Contains mappings for creating, getting, updating and deleting users, which have a one-to-one relationship with Inventory and a one-to-many relationship with Report below. Has the mapping for changing the users high score, getting and setting whether they can chat, how many ban strikes they have, and getting and setting what color they have equipped.

- Inventory: Contains the mapping to get what colors the user has purchased and update what colors the user has purchased.
- Report: Contains the mapping to get all reports, all reports made about a specific user,

PUT THE TABLE RELATIONSHIPS DIAGRAM on this fourth page! (Create the picture using MySQLWorkbench)




report
id BIGINT(20)
content LONGTEXT
reported_message LONGTEXT
reported_username VARCHAR(255)
sent DATETIME
username VARCHAR(255)
reported_user_id INT(11)
Indexes
user
id INT(11)
ban_strikes INT(11)
can_chat BIT(1)
high_score INT(11)
high_score_monthly INT(11)
high_score_time INT(11)
high_score_time_monthly INT(11)
high_score_time_weekly INT(11)
high_score_weekly INT(11)
name VARCHAR(255)
password VARCHAR(255)
selected_color VARCHAR(255)
inventory_id INT(11)
Indexes
user_reports
user_id INT(11)
reports_id BIGINT(20)
Indexes
message
id BIGINT(20)
content LONGTEXT
likes INT(11)
sent DATETIME
user_name VARCHAR(255)
Indexes
inventory
id INT(11)
green BIT(1)
lightblue BIT(1)
magenta BIT(1)
orange BIT(1)
purple BIT(1)
yellow BIT(1)
user_id INT(11)
Indexes