Design Document for MusiQuest

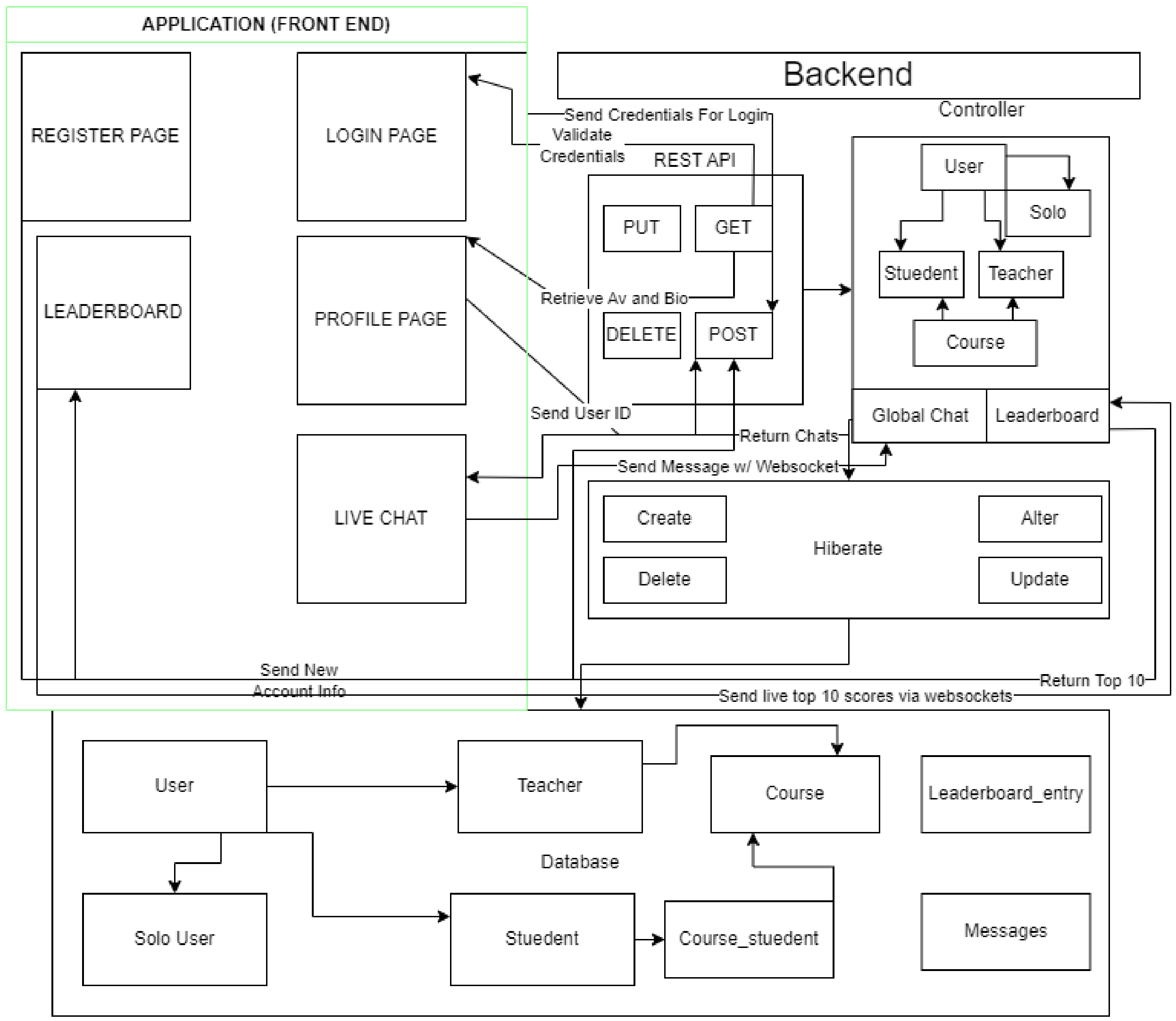
Group ms_311

Gabriel Bullerman: 25%

Haakon Hauswirth: 25%

Jaya Davis: 25%

Tyler Kloser: 25%


APPLICATION (FRONT END)
REGISTER PAGE
LOGIN PAGE
LEADERBOARD
PROFILE PAGE
LIVE CHAT
Backend
Send Credentials For Login
Validate Credentials
REST API
PUT
GET
DELETE
POST
Retrieve Av and Bio
Send User ID
Return Chats
Send Message w/ Websocket
Controller
User
Solo
Stuedent
Teacher
Course
Global Chat
Leaderboard
Hiberate
Create
Delete
Alter
Update
Send New Account Info
Return Top 10
Send live top 10 scores via websockets
User
Teacher
Course
Leaderboard_entry
Database
Solo User
Stuedent
Course_stuedent
Messages

Frontend

Frontend (currently implemented)

Register (SoloPlayer, ClassPlayer, Teacher)

Register generates a page with the following elements:

- EditText: name
- EditText: Password
- EditText: Email
- EditText: Bio
- EditText: profilePicture
- Button: Register

Upon clicking the button 'CreateAccount' the values of the UserID, Password, Email, Bio, and profilePicture are sent as a POST request to the server.

Login (SoloPlayer, ClassPlayer, Teacher)

Create account generates a page with the following elements:

- EditText: name
- EditText: Password
- EditText: Email

Login Screen takes user input of their username and password, and if that username and password is valid, it will take the user to the main menu, where they are logged in as that user.

LeaderBoard(SoloPlayer, ClassPlayer, Teacher)

LeaderBoard generates a page with the following elements:

- Text: Rank
- Text: name
- Text: TotalScore

•Going the this page or clicking the button 'LeaderBoard' completes a GET request from the server into the dynamic table 'leaderboard'

Profile(SoloPlayer, ClassPlayer)

Profile generates a page with the following elements:

- Text: name
- Text: Email
- EditText: Bio
- Image: profilePicture
- Button: Change Bio

Clicking the button 'Account List' completes a POST request to the server with the user ID and the New Bio

Backend

Communication

The backend uses mappings to update the database based on information sent to the given mappings' URLs. These include:

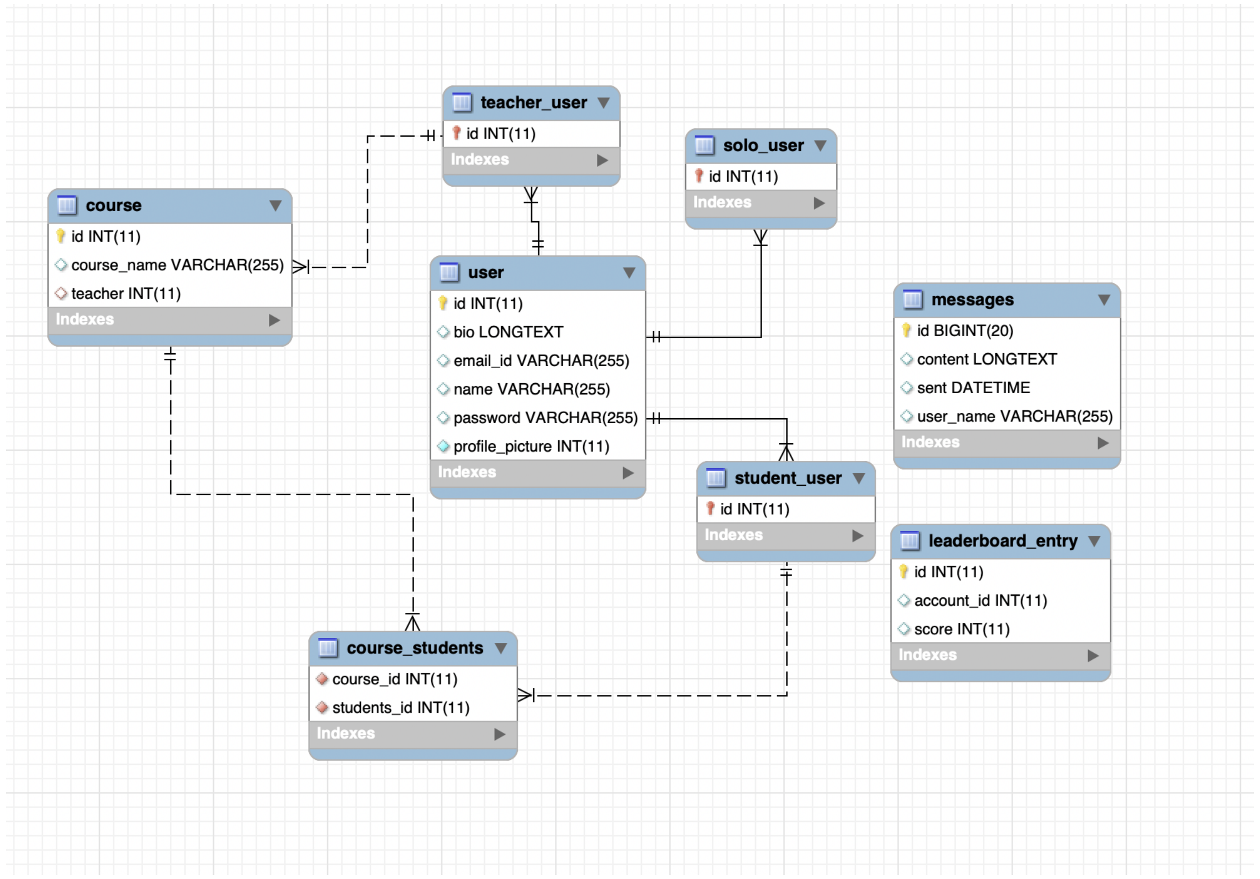
• **Post**: send information on an item to be added to the database.
• **Get**: request information, often with an identifier for the specific item requested from the database
• **Put**: send information to update a specific item in the database
• **Delete**: send an identifier to delete a specific item from the database

Controllers

The controllers contain the mappings for communication between frontend and the database. These include:

• **User**: Contains the above mappings to create users, which contain one-to-one relationships with the below Student User, Teacher User, and Solo User. All User types are sub-Users, all users can also have a score in leaderboardEntry
• **Solo User**: Has no direct relationship with the other types of Users
• **Teacher User**: Can create a course and add students to the course
• **Student User**: Can join a course that belongs to a teacher
• **Courses**: Contains the above mappings to manage courses, a course has a many-to-many relationship with students and a many-to-one relationship with a teacher. Only a teacher can add or remove a student from a course.
• LeaderboardEntry: Contains the above mappings to manage LeaderboardEntry, links a score to a userID, and has been implemented with a web socket that automatically updates when there is a new top 10 score
• Global Chat(Messages): A websocket that allows all users to be able to communicate in one chat. Uses onOpen, onMessage, and onClose. The chat is also being saved

Database Diagram:


teacher_user
id INT(11)
Indexes
solo_user
id INT(11)
Indexes
course
id INT(11)
course_name VARCHAR(255)
teacher INT(11)
Indexes
user
id INT(11)
bio LONGTEXT
email_id VARCHAR(255)
name VARCHAR(255)
password VARCHAR(255)
profile_picture INT(11)
Indexes
messages
id BIGINT(20)
content LONGTEXT
sent DATETIME
user_name VARCHAR(255)
Indexes
student_user
id INT(11)
Indexes
leaderboard_entry
id INT(11)
account_id INT(11)
score INT(11)
Indexes
course_students
course_id INT(11)
students_id INT(11)
Indexes