Design Document for FitQuest

Group <AN_4>

TianyiLuo: % contribution

Taewan Kim: % contribution

Chandrashekar : % Api Documentation and Block Diagram.

Member4 Name: % contribution

Design Descriptions

Android Views

The Android Views contains user interfaces for 8 activities. Each of the activities is connected to a xml layout file as the design of the interface. The activities contain the logic of the user command. But for updating requests to the server, passing user information throughout the application and generating dates each activity would be using the helper classes for these commands.

Frontend (currently implemented)

SignUp(User)

- Create account generates a page with the following elements:
    - EditText: firstName
    - EditText: lastName
    - EditText: email
    - EditText: password
    - Button: SignUp
- Upon clicking the button ‘SignUp’ the values of the firstName, lastName, email, and password are sent as a POST request to the server.

Login(User)

- Login generates a page with the following elements:
    - EditText: Email
    - EditText: password
    - Button: Login
- Login screen takes user input of their username and password, and checks if username and password is valid through json POST request to the server, and if valid, it takes the user to the welcome/main menu, where they are logged in as that user.

FindFriend(User)

- FindFriend generates a page with the following elements:
    - EditText: Email
    - Button: addFriend
- FindFriend screen allows users to input an email address for the friend they wish to add. Upon clicking the button ‘addFriend’ the values of an email address are sent as a POST request to the server.

FriendList(User)

- This class doesn’t require any user input. It displays the list of friends that have been added through the ‘Add Friend’ screen using a json GET method, and lists their email addresses as a Recycler View

Milestone(User)

- Milestone generates a page with the following elements:
    - EditText: workoutNameEditText
    - EditText: weightsEditText

    - EditText: repsEditText
    - EditText: setsEditText
    - Button: addButton
    - Button: progressButton
- Milestone screen allows users to input workoutname, weights, reps and sets for the milestone progress they wish to add. Upon clicking the button ‘addButton’ the values of four data is sent as a POST request to the server. Upon clicking the button ‘progressButton’ takes the user to the milestoneProgress screen.

MilestoneProgress(User)

- This class doesn’t require any user input. It displays the list of milestones that have been added through the ‘milestone’ screen using a json GET method, and lists the milestones as a RecyclerView.

LeaderboardAdd(User)

- LeaderboardAdd generates a page with the following elements:
    - EditText: addpushupsEditText
    - Button: addpushupsButton
- LeaderboardAdd screen allows users to input a number of pushups for the leaderboard progress they wish to add. Upon clicking the button ‘addpushupsButton’ the values of number of pushups data is sent as a POST request to the server.

Leaderboard(User)

- This class doesn’t require any user input. It displays the list of leaderboards that have been added through the ‘leaderboardAdd’ screen using a json GET method. List ranks the user according to the number of push ups. Also has a delete button where you can delete users information if they no longer want to be in the leaderboard.

UserMenu(User)

- Navigate to all other functions
    - Button: findFriendsBtn
    - Button: FriendListButton
    - Button: profileBtn
    - Button: LogOutBtn

    - Button: recordBtn
    - Button: chatBtn
    - Button: milestoneBtn
    - Button: communityBtn
    - Button:leaderboardBtn
    - Button:dietBtn
- Viewing the diet progress and goal.
    - TextView: dailyCalTxt
    - ProgressBar: progressBar

chatPage(User)

- The Android system is outputting Chat messages using RecyclerView.
    - RecyclerView: chatMessagesArray
        - Chat_RecyclerViewAdapter
        - chatModel
- Sending messages to other users.
    - Button:sendChatBtn
    - EditText:chatingMessage

profilePage(User)

- Displaying user profile information of age, gender, weight, username and weight.
    - TextView:genderTxt
    - TextView:ageTxt
    - TextView:emailTxt
    - TextView:weightTxt
    - TextView:userNameTxt
- Navigate to editProfirePage
    - Button:editPBtn

editProfielPager(User)

- Edit profile by the user.
    - EditText:edGenderTxt
    - EditText:edAgeTxt
    - EditText:edEmailTxt
    - EditText:edWeightTxt
    - EditText:edUNTxt

- Send user data to the server.
    - Button:saveBtn

dietPage(User)

- Viewing the diet intake for each day of the week. For each day, there are TextViews of total dietary intake and percentage to the goal.
- Navigate to each day of the week to view all meals recorded.
    - Button: Monday,Tuesday…

DailyDiet(User)

- Displaying meals for the day in a LinearLayout.

DietGoal

- User can edit the daily diet goal in this class.
    - EditText: goalCalTxt
    - Button: setBtn
- Viewing the weekly total as inputting daily goal.
    - TextView weeklyCalTxt

workoutHistoryPage

- Viewing the workout history recorded by the user.
- Navigate back to The main menu.

workoutHistoryAdd

- Edit and record the workout history.
    - EditText: exName
    - EditText: exRep
    - EditText: exMax
    - EditText: exSets
    - Button: saveBtn

BACKEND

Communication:

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

- Post: Mapping to send data from frontend to the backend to add in the database.
- Get: Mapping to request data from the backend by the frontend to be retrieved from the database.
- Put: Mapping to update data which already exists because it was preset or was posted by a post mapping earlier.
- Delete: Mapping to delete specific data from the database sent by the frontend, along with a usually unique or occasionally associated with a single user/object, to the backend.

Controllers:

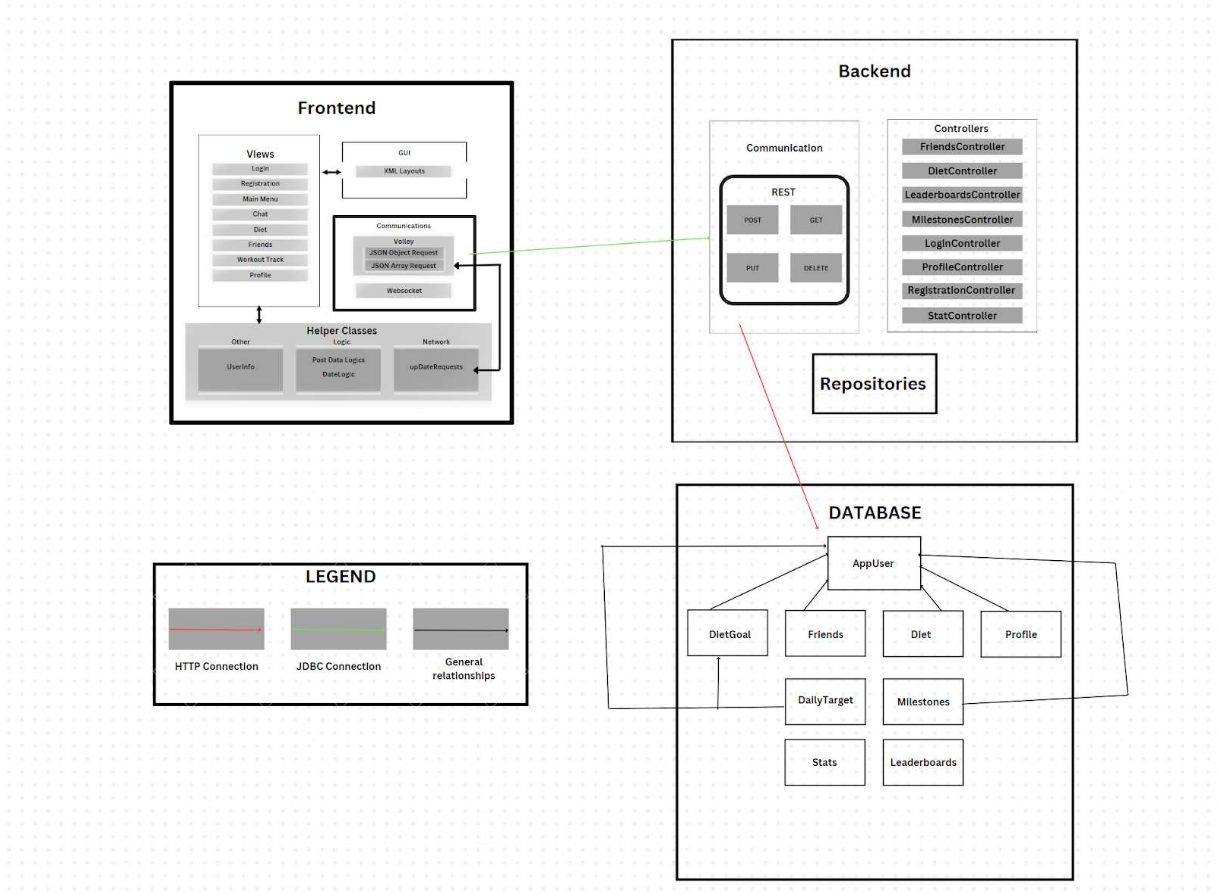
These controllers contain the mapping which the backend uses to communicate with the frontend and accordingly modify/retrieve data from the database. These include:

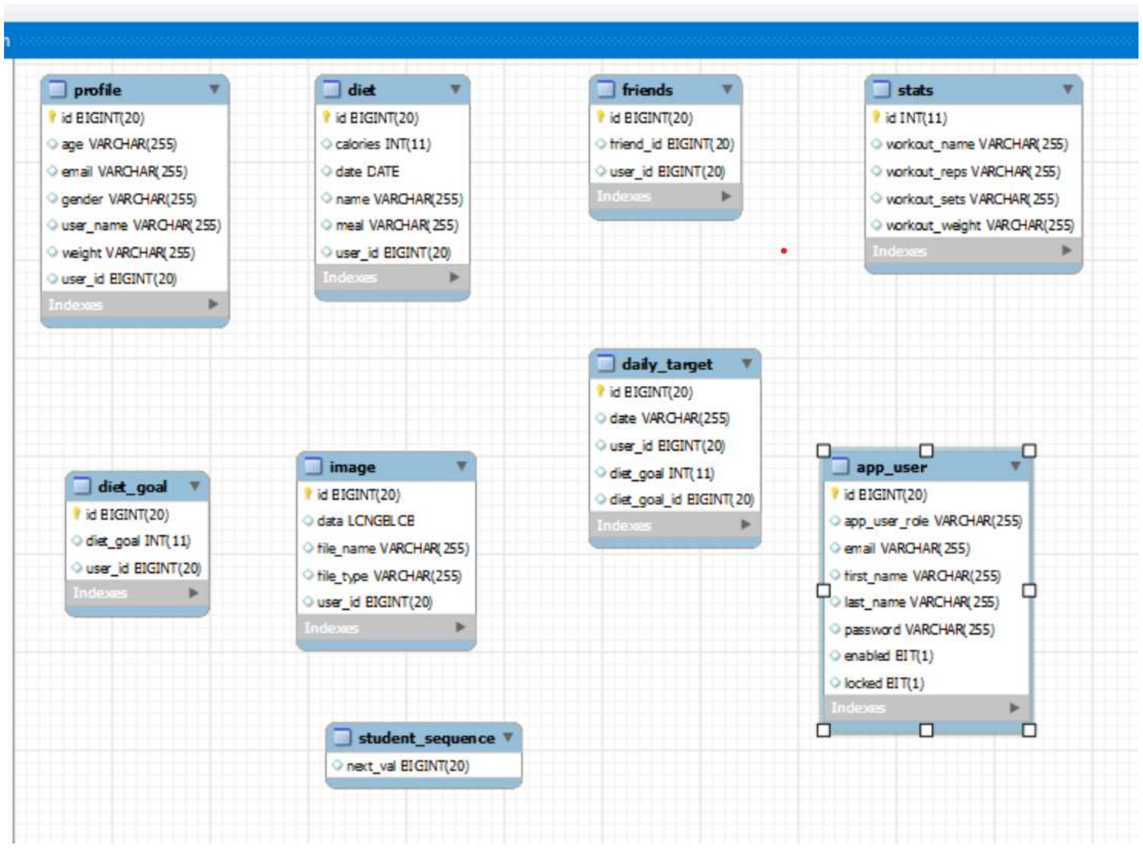
- LoginController: This controller has just one post mapped method.
    - login: This method takes the input sent by the frontend, checks if it matches with the data in the database then returns a positive response if the user exists in the table and the password is right or a negative response if either the user doesn't exist or if the password is wrong.
- RegistrationController: This controller has one post mapped method and a put mapped method.
    - register: This method allows the user to create a new profile on the application and updates the app_users table in the database.
    - updateProfile: This method allows developers to update an already existing profile by modifying its data in the app_users table in the database. This method isn't accessible by the front end and exists only for testing by the backend.

- ProfileController: This controller contains a get mapping, a post mapping and a put mapping.
    - getProfile: This method allows the frontend to request all the data of a profile from the app_users table in the database.
    - createProfile: This method allows the user to create a new profile on the application and updates the app_users table in the database.
    - updateProfile: This method allows users to update their already existing profile by modifying its data in the app_users table in the database.
- StatController: This controller has two get mappings and a post mapping.
    - saveWorkout: This method allows the frontend to send workout data which the backend puts in the stats table in the database.
    - getAllWorkouts: This method lets the frontend request all the workouts to the backend which is stored in the database.
    - findWorkoutById: This is a method for testing on the backend which allows us to access individual workouts using their ID through postman or other means.
- LeaderboardsController: This controller has a post mapping, and two get mappings.
    - saveWorkout: This method allows the frontend to send workout data which the backend puts in the max personal best for one set of pushups in the leaderboards table in the database.
    - getAllUsers: This method lets the frontend request all the users on the leaderboards, sorted largest to smallest by number of pushups, to the backend which is stored in the database.
    - findLeaderboardById: This is a method for testing on the backend which allows us to access individual users along with their pushups using their ID through postman or other means.
- MilestonesController: This controller has a post mapping, a put mapping, two get mappings and a delete mapping.
    - saveMilestone: This method lets the frontend allow users to create a milestone which is by default set to incomplete by creating a new entry in the database.

    - updateMilestoneStatus: This method lets the frontend update the status of the milestone from incomplete to completed, modifying the value of only one field in the database.
    - getAllMilestones: This method allows the frontend to request the backend to retrieve all the milestones of the logged in user from the database.
    - findMilestonesById: This allows the frontend to get the data of a single milestone by its unique ID in the table in the database.
    - deleteMilestone: This allows the frontend to make a delete request to the backend to delete a row from the table in the database by it's unique ID.
- FriendsController: This controller has a post mapping and a get mapping.
    - addFriend: This method lets the frontend send the email ID of the friend the logged in user wants to add. The backend then goes through the app_user table to check for the userID of the email sent by the frontend. If the email doesn't exist, it throws a user not found error. If the user exists but is the user logged in themselves, then it returns "You cannot add yourself as a friend". Otherwise, it adds the userIDs of both the users in the table.
    - getFriends: This method allows the frontend to request the friends of the user who is logged in to the backend.
- DietController: This controller has two post mappings, two put mappings, three get mappings and a delete mapping
    - addDietGoal: This allows the frontend to make a post request to the backend which adds a dietGoal in diet table in the database.
    - updateDietGoal: This allows the frontend to send a put request to the backend which modifies the dietGoal of the user.
    - getDietGoal: This allows the frontend to request all the diet goals associated with the user to the backend which retrieves the data from the database.
    - getDietByUserAndDate: This lets the frontend request the backend for the diet created by the logged in user by time and date.
    - addDiet: This allows the frontend to let the user add their diet to the database.

- updateDiet: This allows the frontend to let the user modify the diet that they have added in the database by modifying the entry in the table.
- deleteDiet: This lets the frontend make a delete mapping which lets the backend delete the entry of a diet in the database.
- getDietsByDate: This allows the frontend to request diet by date to the backend which retrieves this data from the table in the database.





profile
- id BIGINT(20)
- age VARCHAR(255)
- email VARCHAR(255)
- gender VARCHAR(255)
- user_name VARCHAR(255)
- weight VARCHAR(255)
- user_id BIGINT(20)
- Indexes

diet
- id BIGINT(20)
- calories INT(11)
- date DATE
- name VARCHAR(255)
- meal VARCHAR(255)
- user_id BIGINT(20)
- Indexes

friends
- id BIGINT(20)
- friend_id BIGINT(20)
- user_id BIGINT(20)
- Indexes

stats
- id INT(11)
- workout_name VARCHAR(255)
- workout_reps VARCHAR(255)
- workout_sets VARCHAR(255)
- workout_weight VARCHAR(255)
- Indexes

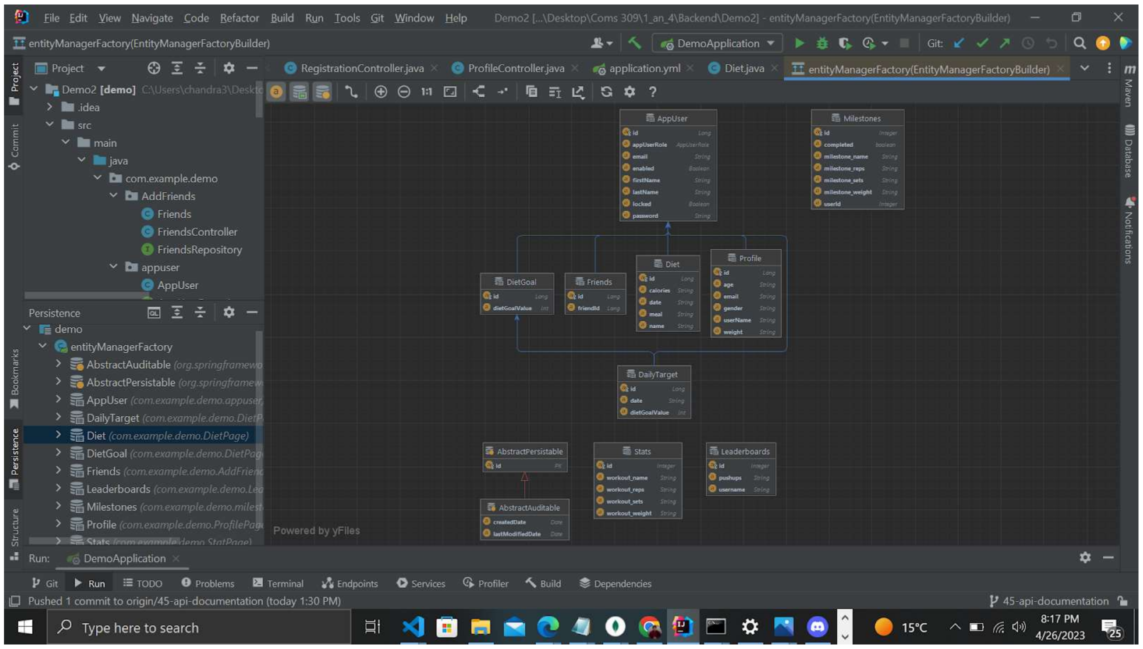
daily_target
- id BIGINT(20)
- date VARCHAR(255)
- user_id BIGINT(20)
- diet_goal INT(11)
- diet_goal_id BIGINT(20)
- Indexes

diet_goal
- id BIGINT(20)
- diet_goal INT(11)
- user_id BIGINT(20)
- Indexes

image
- id BIGINT(20)
- data LONGBLOB
- file_name VARCHAR(255)
- file_type VARCHAR(255)
- user_id BIGINT(20)
- Indexes

app_user
- id BIGINT(20)
- app_user_role VARCHAR(255)
- email VARCHAR(255)
- first_name VARCHAR(255)
- last_name VARCHAR(255)
- password VARCHAR(255)
- enabled BIT(1)
- locked BIT(1)
- Indexes

student_sequence
- next_val BIGINT(20)


Demo2 [...\Desktop\Coms 309\1_an_4\Backend\Demo2] - entityManagerFactory(EntityManagerFactoryBuilder)
RegistrationController.java
ProfileController.java
application.yml
Diet.java
entityManagerFactory(EntityManagerFactoryBuilder)
AppUser
id Long
appUserRole AppUserRole
email String
enabled Boolean
firstName String
lastName String
locked Boolean
password String
Milestones
id Integer
completed boolean
milestone_name String
milestone_reps String
milestone_sets String
milestone_weight String
userId Integer
DietGoal
id Long
dietGoalValue int
Friends
id Long
friendId Long
Diet
id Long
calories String
date String
meal String
name String
Profile
id Long
age String
email String
gender String
userName String
weight String
DailyTarget
id Long
date String
dietGoalValue int
AbstractPersistable
id PK
AbstractAuditable
createdDate Date
lastModifiedDate Date
Stats
id Integer
workout_name String
workout_reps String
workout_sets String
workout_weight String
Leaderboards
id Integer
pushups String
username String
Powered by yFiles
Pushed 1 commit to origin/45-api-documentation (today 1:30 PM)
45-api-documentation
8:17 PM
4/26/2023

API DOCUMENTATION:

coms-309-004.class.las.iastate.edu:8080/swagger-ui.html