Toronto Fitness Club

Due date: Thu Nov 18, 2022 22:00:00 | 00:00:00:00 remaining

Overview

Toronto Fitness Club (TFC) is the new rival for GoodLife Fitness and LA Fitness in Toronto. The Canada-based business owners founded a new chain fitness club in Toronto aiming to expand their business to other major cities in North America. A successful business requires smart investment, perspective, and planning. It is why the business owners understand the significance of a having perfect website from the very first stages. They hired you to build their website. For some reason, the owners lack patience, so you need to deliver the website very quickly. In return, they are going to give you life-time free admission to all yoga classes, group meditation, and therapy sessions held by TFC.

The project is divided into **two parts**. Firstly, you will use Django and REST Framework to implement the backend of the project.

Course Links

- Announcement - Q&A: Piazza
- Submissions: Markus
- Informal chat: Discord
- Interviews and grades: IBS
- Lecture codes: Github

Secondly, you will build the frontend using React JS and connect the backend to it. The eventual outcome of this project is a fully-functional website that meets the **user stories** listed in an upcoming section.

The project is handed out from the business owner's point of view. That is, those people do not necessarily have a computer science background. Therefore, the requirements are described in a non-technical way, and it is your job to design and implement the web application that meets them. There is much room for creativity, and you get decent freedom with respect to the details and various tools you can use for this project. In fact, there are very few general limitations. For example, you will have to use **Django, REST, and React**. Beside these, you can utilize any online packages, library, or open-source code and template to improve your website.

Deadline

This part is due on **Friday, Nov 18, 2022 at 10:00 PM**. Late submissions of x hours (starting from 1) will deduct following percentage from your group's grades.

$$p = 0.0002x^{3.06}$$

Academic Integrity

Unlike the assignments, you are allowed to download packages or use open source codes from the internet for the project. However, **sharing or publishing** even a small piece of code to

other teams is strictly prohibited (either giving to or taking from them). Online codes must include a reference to the webpage they are taken from.

Groups

Register your groups on Markus as soon as you have found the teammates. The scope of the project is appropriate for a group of three members. However, if some members have already had prior knowledge in web development, groups of two would be fine as well. You can also choose to work entirely on your own, which is strongly discouraged as the load will be overwhelming. If you have a solid background in web development and want to intensify and challenge your knowledge, working alone could be an option.

Note: The number of members will not affect our grading criteria. That is, your submission will be graded the same way, regardless of how many team members are there.

Mentor sessions

For each part, you can sign up for two mentor sessions with a TA to discuss your ideas, ask questions, or seek help on any project-related issue. You must already have registered your teams on Markus and use the IBS to sign up for a session. Note that sessions have an availability interval. For example, the first mentor session is available only in specific dates (e.g., two weeks before the deadline). Therefore, if you do not book one, you will lose one session and will not get two sessions in the next week. More details are explained on the IBS docs.

Interviews

Each part will be graded separately through interview sessions with TAs. Interviews are unsurprisingly booked through the IBS system (after all, that is why it is called IBS in the first place). Please note that if you do not book a session, you will get **no marks** for that part. All team members must be present at the meeting, and the absent members will not get any marks for the part. Time intervals associated with the mentor sessions and interviews will be announced on Piazza.

Submission

You should submit your work to your group's git repository on Markus. That is, you cannot submit through its normal web interface. Instead, commit and push your codes to your repository for each part. Find out your repo's URL by logging into Markus. Note that there is a separate repository for each part.

Moreover, commit your changes now and then, and do not leave your commits to the **very last minutes** before the deadline. Correct use of git is a part of your assessment for this course.

Environment

Your code must run in an Ubuntu 20.04 machine that has python 3.10 and virtualenv already installed. It is fine if you are developing your code on other environments, although Windows is not encouraged. However, you **must** make sure that

your server and scripts (described later) run without any problem on Ubuntu 20.04 before submitting. The teach.cs accounts are good examples of such environments.

Participation

Each team member must do a fair amount of work on this project. We will assess each member's participation by investigating the commits they created and asking detailed questions about the code at the interviews. Even though it is a group project, your final grade is **individual**.

Project description

The following paragraphs describe the features that should be included in TFC. As stated earlier, features are explained in terms of user stories, in a non-technical form, and from an imaginary user's point of view. You should not expect the same level of detail as assignments. Details that are not discussed below are left out to you, and you get to design them. Your final grading will be done mostly based on the overall functionality of your website as if the business owner is testing your application.

User stories

Accounts

> As a user, I can sign up, log in, log out, and edit my profile. Profile information should include first and last name,

email, avatar, and phone number.

Studios

As the **website admin** (i.e., an account that has access to the admin panel), I can create/edit/delete studios. A studio has a name, address, geographical location, postal code, phone number, and a set of images.

As the **website admin**, I can update the amenities of a studio (type and quantity).

As a user, I want to list studios based on geographical proximity to a specific location (e.g., my current location, a pinpoint on map, or a postal code). Studios must list starting from the closest one, each having a drop pin on the map.

As a user, I can click on each of the studios and move to the studio page. This page should contain the general information of that studio, along with its address, location, and a link to get the directions.

Classes

As the **website admin**, I can create/edit a class in a specific studio. A class has a name, description, coach, a list of keywords (e.g., upper-body, core, etc.), capacity, and times. Times indicate the recurring instances of the class. For example, a HIIT session is held on Mondays from 8:00-9:00am.

Update: To simplify the concept of recurring classes, your class can now have a mandatory "end recursion" date. For example, a HIIT session will be held each Monday at 8am until December 1, 2022. However, if your design allows for classes to recur **indefinitely**, (and works well with the user stories), there will be a bonus.

As the **website admin**, I can cancel a specific class; either one instance or all the future occurrences of the class.

As a user, I want to see the class schedule of a specific studio on its page. Classes must appear in the order of their start time (from now), and the class information must be shown. Past or cancelled classes should not be listed.

As a user, I can enrol/drop a class (either one instance or all future occurrences) that has not started yet and has not reached its capacity. This can only happen if I have an active subscription.

As a user, I want to see my class schedule and history in chronological order

Searching and filtering

As a user, I want to search/filter through the listed studios (mentioned earlier). Search/filter includes stdio name, amenities, class names, and coaches that hold classes in those studios.

As a user, I want to search/filter a studio's class schedule. The search/filter can be based on the class name, coach name, date, and time range.

Subscriptions

As the **website admin** I can create/edit/delete the gym subscription plans (e.g., 14.99$ per month or $149.99 per year).

As a user, I can subscribe to one of the options; the first payment is made immediately after subscription, and each upcoming payment will be automatically made in the beginning of their period.

As a user, I can update my credit/debit card information. All subsequent payments must be made to the updated card.

As a user, I can see my payment history (amount, card info, date and time, etc.), as well as my future payments.

As a user, I can cancel or update my current subscription. The next payment will be either cancelled or updated. In case of cancellation, all class booking after the current billing period will be invalid.

Part 1 deliverables

In this part, you will implement the Django backend for the

provided in the admin panel. Django's admin panel is so powerful, and you will be able to implement all the admin user stories using it. All other user stories are implemented with one or more REST APIs. Your APIs should not have any template or HTML response.

The project should run in a **virtual environment**, with all the packages listed in the requirements file. Ideally, your backend server should not go through many changes at the next part (this will not happen in practice).

You should implement proper authentication and authorization for the user stories as well.

Note 1: As potentially ambiguous as the handout gets, you can ask your questions on Piazza or discuss them with TAs at mentor sessions.

Note 2: It is recommended that you replace the Django's default User model with a custom model that meets the requirements of this handout. Your new model can extend the default one, or have a one-to-one relation with it. These instructions helps Django recognize your custom model as the auth model so that all User-specific functionalities (sessions, tokens, etc.) still work.

Note 3: All API views that return a potentially long list of items (e.g., list of studios, classes, comments, etc.) must be **paginated** so that a reasonable amount of data is returned in one response. Read more information here.

What to submit

You should push your entire Django project to your repo, accompanied by a startup.sh script, a run.sh script, a docs.pdf file, and a postman.json collection. These files should be located in the root folder of your repository.

The **startup** should run any preparation needed for your code to run in a new environment. It should create the virtual environment, install all required packages with pip, and run all migrations. The **run** script should start your server. Finally, your documentation must include your design of models, as well as the full list of all API endpoints, a short **description**, their methods, and the payloads. You can use packages that automatically generate the API docs from your code. The TA will send requests based on the information you provide on that document. Therefore, it is important to have a usable and clear document. The **postman** collection must be an export of all the

APIs (not the admin panel views) that are importable into Postman. The TA will use Postman to test your APIs, so it is important that your collection is comprehensive, well-organized, and self-explanatory. All the POST data, headers, and GET query parameters must be pre-filled and editable in that collection.

Note: Before submitting, make sure that your startup and run scripts work well on the virtual machines given to you, and your collection can be imported into Postman without any issues. The TAs will run your application on clean instances of that virtual machine. If you have worked with another operating system, it is your absolute **responsibility** to double-check that your project works fine within the virtual machine as well.

Comments

Large Brain ★★★★★

Mon Dec 12 2022

This is my seventh time taking CSC309, and inarguably, this is the best assignment I've ever seen.
Problems are so accurately designed that anyone with their own knowledge and experience have their own takeaways.
I hope next assignments will be this good, too!

108 | Reply

Kianoosh

Mon Dec 12 2022

Thanks bro!

36 | Reply

Captain FiFi ★★★★★ ★★★★★

Hey may I suggest that I implement the actual functionality of the button below in exchange for 10% bonus of the final grade? Collecting people's feedback can help promote democracy, equity, and freedom of speech. A diverse environment where every voice is heard is conducive towards fostering a better post-secondary education. Alternatively, I can charge you for $500.

83710 | Reply

Write a comment

Your email address will not be published. Required fields are marked *

Review*

Name*

Email*

Notify me of new posts by email

SUBMIT