CS 1XA3 Project 03

Due Date: April 23 2019 at 11:59pm

Contents

1 Project Setup

1.1 Add The Project03 Template To Your Repo

- You can download the starting template project from `https://github.com/dalvescb/django_templates`, under Project03
- Simply copy the Project03 folder in its entirety into **your own github repo**, i.e so there now exists a folder `CS1XA3/Project03`

1.2 Replace macid with you're Mac ID (necessary to run on mac1xa3.ca)

- Search and replace instances of macid with you're actually McMaster ID in the following files
  - `Project03/Project03/urls.py`
  - `Project03/Project03/settings.py`

1.3 Commit and Push

- Add and commit this folder (to the master branch) with the following message EXACTLY "Initial Project03 Commit"
- Push (to the master branch) to GitHub

1.4 Create a Branch

- Create a branch called project03
- Push the branch to github (i.e git push origin project03)
- Work from the project03 branch and only merge with master when your ready to submit

1.5 Work From Your Private Directory When On mac1xa3.ca

- NOTE I highly recommend completing this project from your local machine, and then testing it on mac1xa3.ca when you are finished. However the project **MUST RUN FROM MAC1XA3.CA** as this is where it will be marked, so do not wait too close to the due date to make sure it works there
- When working/running on the mac1xa3 server, you must keep your repo inside of the directory located in `$HOME/private` on the `mac1xa3.ca` server
- WARNING not doing so or changing the default permissions on the private directory will be considered academic dishonesty

1.6 Submission

- When you are ready to submit, merge your project03 branch with master
- Make sure you push to Github
- Make sure an up to date version of your repo is cloned on the mac1xa3.ca server (i.e in `$HOME/private/CS1XA3`)

1.7 Running the Server

- Without completing any of the objectives, you can run the server as is on your local machine with

  ```
  python manage.py runserver localhost:8000
  ```
- then going to localhost:8000/e/macid/ in your browser
- For instructions on running the server on mac1xa3.ca, see the README on `https://github.com/dalvescb/django_templates/tree/master/user_models_2`

2 Project Overview

The goal of this project is to complete the social media like site provided in the Project03 template. The site consists of the following pages:

- login app
  - login page (`Project03/login/templates/login.djhtml`)
  - signup page (`Project03/login/templates/signup.djhtml`)
- social app
  - account settings page (`Project03/social/templates/account.djhtml`)
  - messages display page (`Project03/social/templates/messages.djhtml`)
  - people/friend lookup page (`Project03/social/templates/people.djhtml`)

2.1 Project Database

All of the models (i.e the database) needed for the website have already been created in

- `Project03/social/models.py`

```
class Interest(models.Model):
    # Related to UserInfo as m2m relation, holds interests as strings
class UserInfoManager(models.Manager):
    # implements the create_user_info method for creating UserInfo entries
class UserInfo(models.Model):
    # A 1-to-1 relation with the built-in User objects, adds additional info
class Post(models.Model):
    # holds posts made by users and displayed in the messages page
class FriendRequest(models.Model):
    # holds friend requests from one User to another
```

2.2 Project Templates

The entire project makes use of a base template base.djhtml in `Project03/templates/base.djhtml` that extends the w3schools social media template. By default, i.e un-extended, it provides.

- links to a variety of w3schools stylesheets and meta page definitions (use the css block to extend)
- an import link for JQuery
- a navbar with an icon on the right that links to the account settings page (use the navbar_contents block to extend)
- a footer
- three empty column sections (use the left, middle and right column blocks to extend)

The base template is extending by

- login.djhtml (implements middle_column to provide login form)
- signup.djhtml (should implement middle_column to provide signup form)
- social_base.djhtml (implements left_column to provide account info, and navbar to provide links to account, messages, people pages)

The social_base.djhtml template implements the right_column block and is extended by

- account.djhtml (should implement middle / right columns to provide account settings forms)
- messages.djhtml (should implement middle / right columns to show posts / friends list respectively)
- people.djhtml (should implement middle / right columns to show unfriended people / friend requests respectively)

3 Project Objectives

Perform each of the following objectives to complete the project. Most objectives provide a fair amount of guidance, but you may delete/add as much code as you like to achieve the attended effect of the objective.

- TIP: Follow the TODO Objective N comments in the codebase

3.1 Objective 1: Complete Login and SignUp Pages (5 points)

- There are login and signup pages already set up under the `Project03/login` app
- By default, the URL `/e/macid/` will route to the login
- In order to login you need to create a UserInfo entry (NOTE it is not sufficient to create just a User object)
- You can create a new {{{color(blue,UserInfo)}} object manually from the shell by running the commands (remember to run migrations first)

```
python manage.py shell
    >>> from social import models
    >>> models.UserInfo.objects.create_user_info(username='TestUser',password='1234')
    >>> exit
```

- The login form is already completed, click the Login button to try logging in, then click the Logout button in the Navbar to logout
- If you click the Create An Account button, you'll be redirected to a mostly blank page
- Implement this page by:
  - Add the appropriate code to the function def signup_view in `Project03/login/views.py` to render signup.djhtml with an appropriate form
  - Add the appropriate code in `Project03/login/templates/signup.djhtml` to display a form for creating a new user
  - Add appropriate code to handle the POST request sent by the form to create a new user (i.e automate the code above for creating a UserInfo object)
  - After creating a new user, automatically log the user in and redirect to the messages page

3.2 Objective 2: Adding User Profile and Interests (5 points)

- The template social_base.djhtml renders the left_column used by messages.djhtml,people.djhtml and account.djhtml
- When each of the above templates are rendered in `Project03/social/views.py` (see messages_view, people_view and account_view respectively), they are given the currently logged in UserInfo object via a variable (and hence social_base.djhtml has access to it as well)
- social_base.djhtml currently displays a statically included fake Profile and Interests
- Edit `Project03/social/templates/social_base.djhtml` to implement a real Profile and Interests corresponding to the currently logged in user by using Django Template Variables
- See the UserInfo class in `Project03/social/models.py` for details on the attributes available

3.3 Objective 3: Account Settings Page (10 points)

- Clicking the top right icon on the Navbar brings you to the Account Settings Page, rendered by
  - the function def account_view in `Project03/social/views.py`
  - the template `Project03/social/templates/account.djhtml`
- Currently the page is mostly blank (besides the content already rendered by the base templates)
- Edit the above function / template to:
  - Provide forms for changing the users current password
  - Provide forms for updating user info, i.e employment, location, birthday, interests (each interest submission should add to the list of current interests)
  - Handle POST requests sent by the form's to update the UserInfo object accordingly

3.4 Objective 4: Displaying People List (10 points)

- Clicking the people icon on the Navbar brings you to the People Page, rendered by
  - the function def people_view in `Project03/social/views.py`
  - the template `Project03/social/templates/people.djhtml`
  - the javascript code `Project03/social/static/people.js`
- Currently the middle column displays a single static fake person and a More button
- Edit the above function / template to:
  - Display actual User's in the middle column who ARE NOT FRIENDS of the current user (HINT use a for loop in the people.djhtml template to replicate the current div)
  - The page should start out displaying only a certain amount of people (say 1), then display more by clicking the More button. The amount of people displayed should reset when the user logs out
  - Note: the more button is already linked to send an AJAX POST through javascript (see `Project03/social/static/people.js`) to def more_ppl_view, and then reload the page on success
  - HINT use a session variable to keep track of how many people to display

3.5 Objective 5: Sending Friend Requests (10 points)

- The right column of people.djhtml should display friend requests to the current user (currently only displays a single static fake friend request)
- All Friend Request buttons are linked to a JQuery event in people.js, which uses its id to send a POST request to the function def friend_request_view
- Edit the people.djhtml template to configure the Friend Request button so it's id contains the user who sent the friend request
- Edit the function friend_request_view to handle the POST by inserting an appropriate entry to the FriendRequest model
- Now edit def people_view and people.djhtml to render actual friend requests

3.6 Objective 6: Accepting / Declining Friend Requests (10 points)

- The Friend Requests displayed in Objective 5 contain Accept and Decline buttons
- Edit people.djhtml so that the id's of those buttons contain the username of the user who sent the Friend Request
- Edit people.js so that pushing either Accept or Decline buttons sends a POST to accept_decline_view with the appropriate button id
  - HINT: copy how the friendRequest function works, it will be very similar
- Edit accept_decline_view to handle the POST request, it should delete the corresponding FriendRequest entry, and if the request was accepted should update BOTH USERS friends relation in the UserInfo table

3.7 Objective 7: Displaying Friends (5 points)

- Clicking the message icon on the Navbar brings you to the Message Page, rendered by
  - the function def messages_view in `Project03/social/views.py`
  - the template `Project03/social/templates/messages.djhtml`
  - the javascript code `Project03/social/static/messages.js`
- Currently, the right column shows only a single static fake friend
- Edit messages.djhtml to display all of the friends of the current user
  - HINT: iterative over the objects returned by all of the friends relation in the user_info variable

3.8 Objective 8: Submitting Posts (10 points)

- The top of the middle column of the Messages Page contains a text field (id post-text) and button (id post-button for submitting posts
- Edit messages.js to submit a AJAX POST request when post-button is clicked, sending the contents of post-text to post_submit_view
- Reload the page upon a success response
- Edit post_submit_view to handle the post submission, by adding a new entry to the Post model

3.9 Objective 9: Displaying Post List (10 points)

- Currently the Messages Page displays only a single static fake Post in the middle column (under the submit post div)
- Edit messages.djhtml to display real posts given by messages_view when rendering the template
- Edit the function messages_view to query for posts, sort posts by newest and to oldest
  - HINT: use order_by on the primary key)
- The page should start out displaying only a certain amount of Posts (say 1) and should display incrementally more as the More button is clicked (just like in Objective 4)
- The More button is already configured to send a post to more_post_view
  - HINT use a session variable to keep track of how may posts are displayed
- Reset how many posts are displayed when the User logs out

3.10 Objective 10: Liking Posts (and Displaying Like Count) (10 points)

- Each post has a Like button and should display a Like Count (currently it always displays 0)
- Update the codebase to actually allows users to like posts and display a real like count
- Prevent users from double-liking a post (i.e each user can only like a post once)
  - HINT roughly copy how adding Friend Requests worked (i.e assigning id's that correspond to which friend request was clicked, submitting AJAX POSTs with those id's)
  - DOUBLE HINT when returning posts in message_view, add an extra boolean attribute to each post object that let's you know if the post has already been liked by the user
  - TRIPLE HINT if a post has already been liked, set the button class to w3-disabled and remove the like-button class so the JQuery event no longer responds to it

3.11 Objective 11: Create a Test Database (5 points)

- Create a variety of test users, create many posts and likes and different friend requests to showcase all the functionality you've implemented
- Make sure to add and commit your sqlite3.db file and migrations folders so that the TA's marking your assignment have access to the database you've populated

4 README (10 points)

You MUST document your code in a file `CS1XA3/Project03/README.md`

- The document should be styled with MarkDown (see https://guides.github.com/features/mastering-markdown/)
- The document should describe usage of the server, locally and on mac1xa3, and how to log in with any TestUser you've set up
- The document should contain a section (header) for each objective that gives a description of how you implemented the objective and how you handled any exceptional use cases

An example of the general outline of the document would be as follows (this is not an exact template you need to follow, you are encouraged to use your best judgment for constructing a useful README):

```
#  CS 1XA3 Project03 - <MyMacId>

## Usage
   Install conda enivornment with ...
   ...
   Run locally with
      python manage.py runserver localhost:8000
   Run on mac1xa3.ca with
      python manage.py runserver localhost:100192
   ...
   Log in with TestUser, password SomePassword
   ...
## Objective 01
 Description:
         - this feature is displayed in something.djhtml which is rendered by
           some_view
         - it makes a POST Request to from something.js to /e/macid/something_post
           which is handled by someting_post_view
 Exceptions:
         - If the /e/macid/something_post is called without arguments is redirects
           to login.djhtml
## Objective 02
  ...
```

5 Grading Scheme

README Documentation	**10%**
Test Database	**5%**
Objectives	**85%**

WARNING failure to properly follow instructions (including not cloning your repo to the proper directory, not pushing to GitHub, not using the correct commit message, etc) will result in A MARK OF 0

5.1 Criteria: README Documentation

- 30% for using good style (i.e using proper markdown, proper sectioning of functionality etc)
- 30% for specifying execution/usage information correctly
- 40% for proper detail

5.2 Plagiarism / Academic Dishonesty

Tools will be used to compare your code with your peers

- Do not use anyone's code but your own for this project
- Do not share your code (even for debugging purposes) over Discord (or anywhere else)
- Any account of plagiarism will result in an automatic grade of 0