grade: 70
comments: all diagrams are wrong, need to seriously improve them to do fine in the final exam or future diagramming as a professional. this report did provide some design efforts, hope your team can end up well.

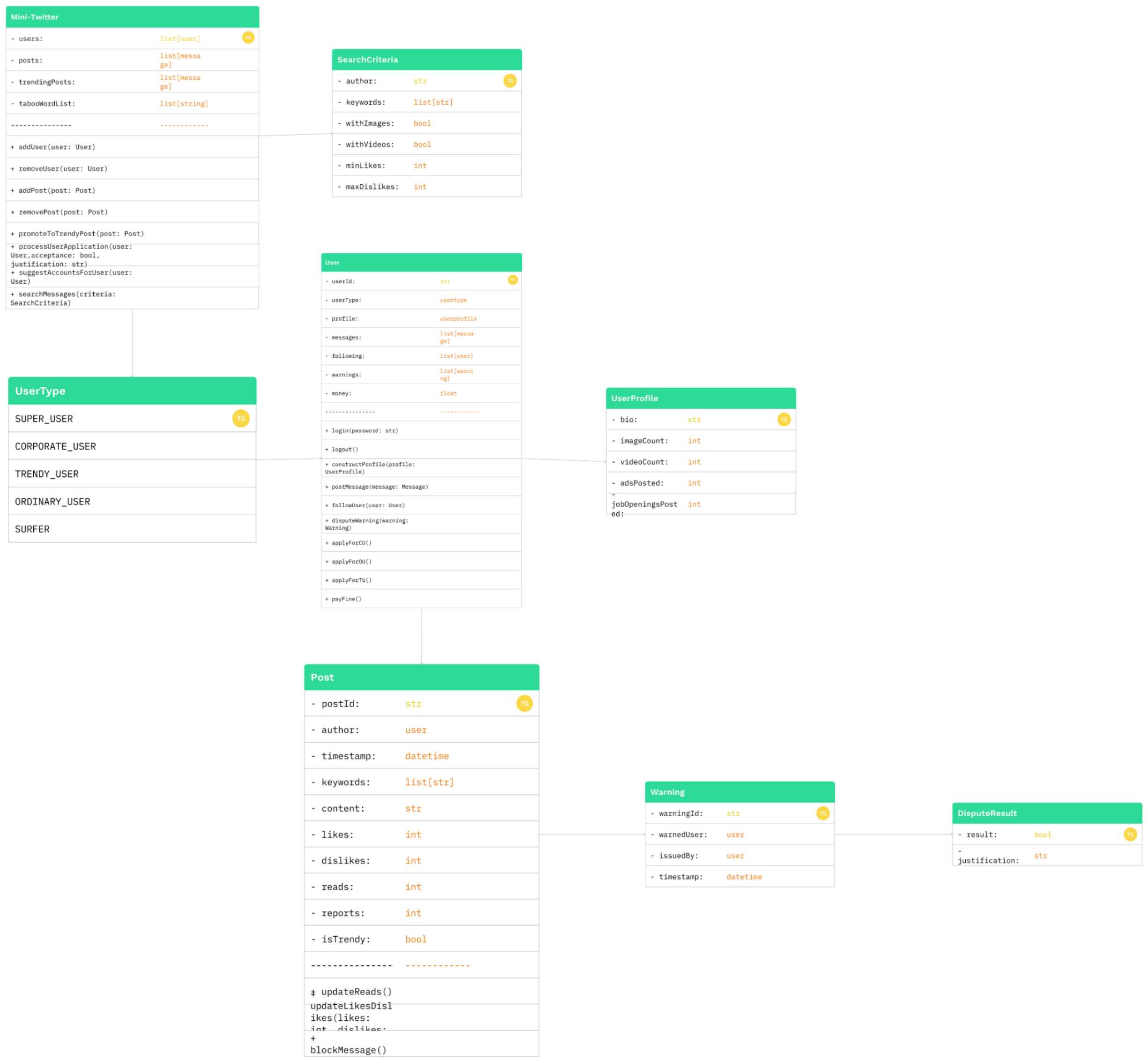
Joquanna Scott, Devin Munoz

Phase II: Design Report

This report is meant to provide the data structure and logic to carry out the functionalities dictated by the specification. Keep in mind the implementation will be entirely based on this document, adequate details should thus be given at length.

1. Introduction:

An overall picture of the system using collaboration class diagram:



this is not the col/seq class diag to describe the system with required details, more like an over-simplified e/r diag.

This collaboration class diagram outlines the structural relationships and interactions between key classes in our Mini-Twitter application. Users are categorized into either a SU (Super User), TU (Trendy User), CU (Corporate User), and/or a surfer, each inheriting common attributes from our User class. Some of these attributes include username, email, and password. In our Post class, users are given the ability to do actions such as posting and updating their profiles. Every post has attributes such as the author, timestamp, and number of likes/dislikes. SUs have special privileges such as handling new user applications and dealing with warnings/complaints. The entire application is built to handle around 10 user accounts along with their posts/comments. More features include promoting messages to "trendy" posts and suggestions for other users to follow. The diagram serves as a comprehensive guide for developing our mini-twitter application, which encompasses all user interactions, their relationships, and system functionalities.

2. All use cases:

o Scenarios for each use case: normal AND exceptional scenarios

- **Applying for an account:**
  - A surfer must submit an application to a Super User (SU) in order to become an Ordinary User (OU) or a Corporate User (CU).
  - A SU will review the application and determine whether to accept or decline.
  - If the application is accepted, the new user is sent a temporary password that they must change on login. Also, the user is expected to commit a deposit to the system.
  - If the application is declined, the SU must provide a justification. A surfer can have multiple attempts at creating an account.
- **Accepting/Declining Applications:**
  - This feature is only available to SUs.
  - SUs will be notified when new applications are pending review.
  - If the SU chooses to accept an application, no more work is done on their part. The temporary password is created and sent by the system.
  - If the SU chooses to decline an application, they must provide a justification that will be sent to the email provided on the application form.
- **Logging In:**
  - A user with an approved account may login with their username and password.
  - If the user makes an invalid input in the username or password fields they will have a warning indicator flashed. This will continue until they enter the correct username, password pair.
  - If the user can not remember their password they can access the reset password form through the Login page.
  - Upon successful login, the user will be directed to the home page of the system.
- **View Suggested Accounts:**

- Upon login, registered users will be shown suggested users to follow.
- The suggested users are based on the like, follow, and tip history of the user. Without an established history, the suggested accounts are based on the current most popular Trendy Users (TUs) and posts.
- The home page of surfers contains the top 3 most liked messages and the top 3 TUs. This layout will be the default for users without a history to base their suggestions off of.

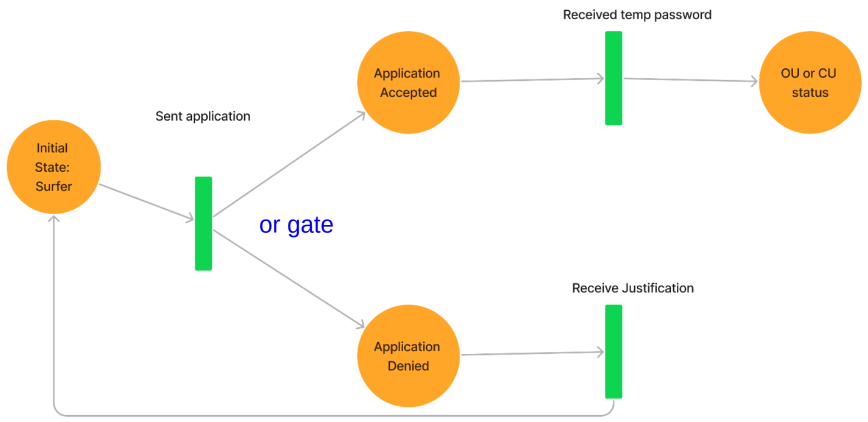
- **Customize Profile:**
  - All registered users have the ability to customize their accounts, including their profile picture, username, biography, and or email.
  - Profiles are subject to comments, likes, and or warnings from other users.
  - In the case of receiving a warning due to profile content, the user will need to change the problematic element or dispute the warning with a SU.
  - If the user desires to change their password, then they must submit a reset password form in order to do so.
  - If a user is not logged in or surfer gains access to the link to the account page, they will be redirected to the login page.
- **Search for Messages:**
  - Any user, registered or surfer, may search for a message based on keywords, author, likes, and or media content.
  - Posts that have been reported for misinformation or have received at least 3 reports will not be shown in the search results. They may be returned to the results after a successful dispute with SU.
- **Post Messages/Comments:**
  - Only registered users are allowed to make posts or comments. Attempts to do so by a surfer will redirect them to the login page.
  - TUs and OUs have the ability to create messages that are equivalent to at most 20 words for free. Any more words will result in an automatic withdrawal from their account balance.
  - CUs must pay for any message posted.
  - Any user who posts a message without the adequate amount in their balance will receive a warning.
  - If the user adds enough money to cover their due balance then the warning will be removed.
  - If the user completes their current session without paying the due balance, upon next login they will be redirected to the payment page.
- **Report Profiles/Posts/Comments:**
  - All registered users have the ability to report another user for misinformation, harmful content, and or other policy breaking material.
  - A user who has received at least 3 reports will be given a warning.

    - After 3 warnings, OUs and CUs have the ability to choose to pay a fee to remove them or allow themselves to be removed from the system. A TU with 3 warnings is demoted to an OU before being given the choice between fine and removal.
    - Any user may try to dispute a warning with a SU, which will result in dismissal or removal of the warning.
- **Issue/Resolve Warnings:**
    - A SU can issue a warning upon being notified that a user has been reported at least 3 times.
    - If a user attempts to dispute a warning, a SU must review the subject of the reports and determine their validity.
    - If the SU decides to remove an outstanding warning, the registered users who made the reports will be warned. If a surfer's report initiated the warning, then the reported user will be given 3 likes.
    - If the SU decides the warning is valid, then they must provide justification to the warned user.
- **Post Ads/Jobs**:
    - Only CUs have valid access to this feature. Other users will be fined $10 and receive a warning.
    - The total cost of a job or ad posting is the default price of a post plus $0.1 for every interaction.
    - If a CU can no longer provide the money necessary to keep up with interactions, they will receive a warning and be directed to the payment page upon their next login.
    - If the CU is unable or unwilling to pay to keep the ad up then it will no longer be displayed.
- **Change Theme:**
    - Registered users will be able to choose a one of 3 premade themes to personalize their view of the system.

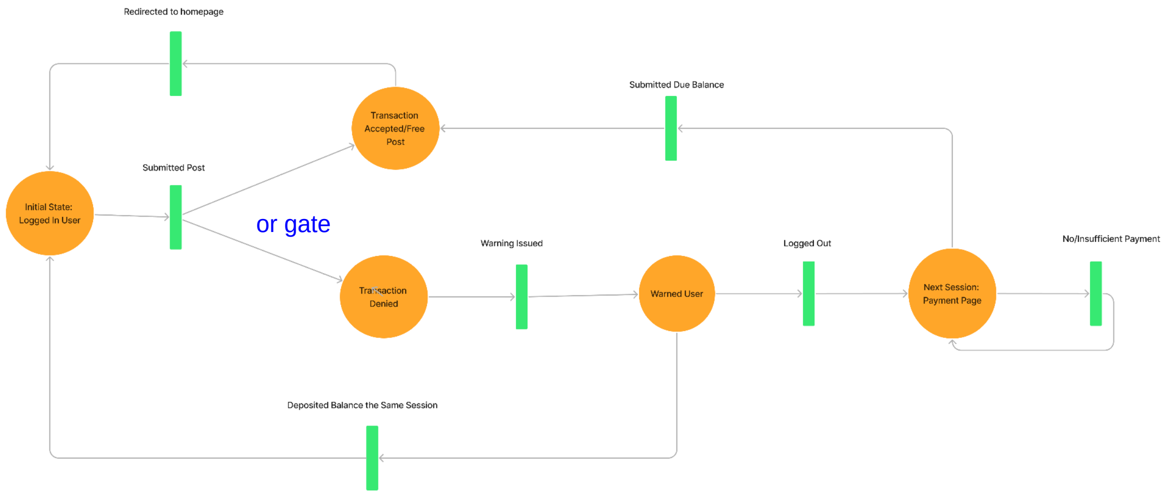
- **Shown top 3 most liked posts and top 3 TUs**
    - Surfers who visit the system are shown the top 3 posts with the most amount of likes and the top 3 users that are most trendy
    - Only users who are not logged in, or Surfers, are only shown this content
    - While surfers have the ability to view these posts, they cannot like the posts shown or report them

o Collaboration or sequence class diagram for each use case, choose
3 or more use cases: draw the Petri-nets instead of class diagrams

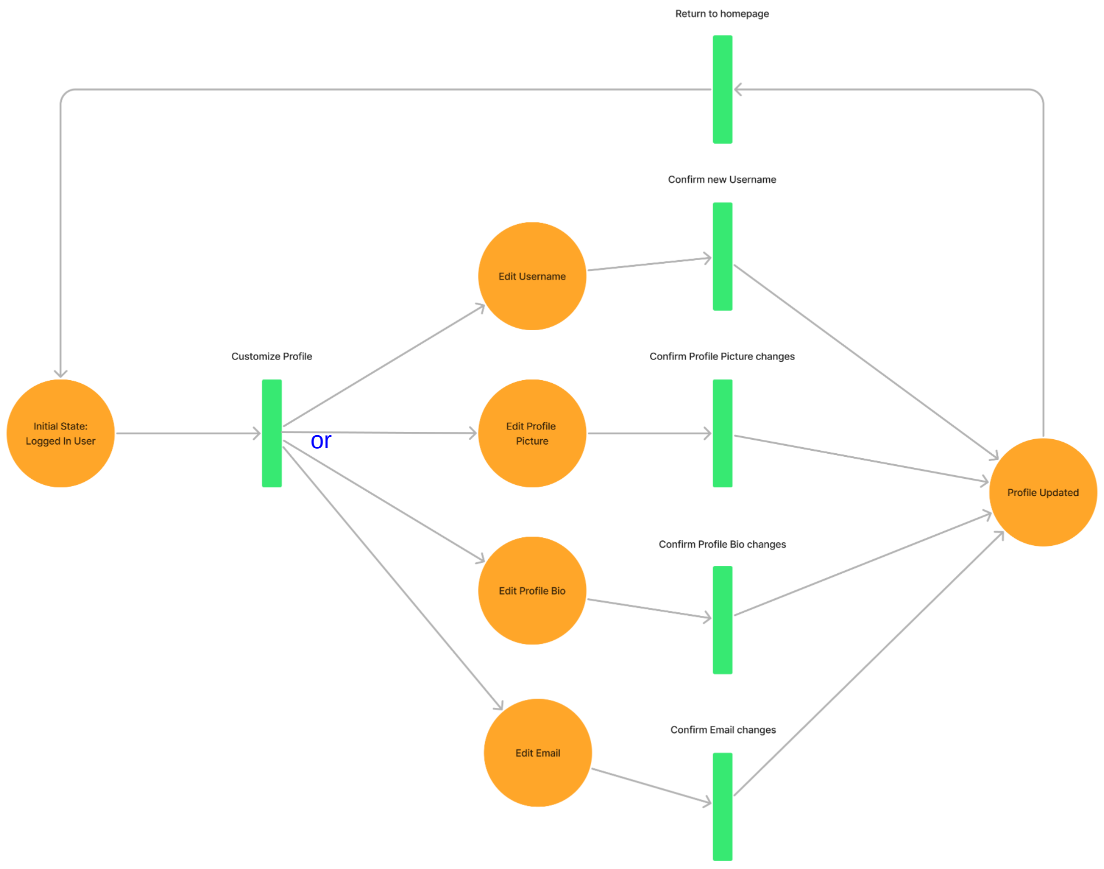
- Applying for an account:



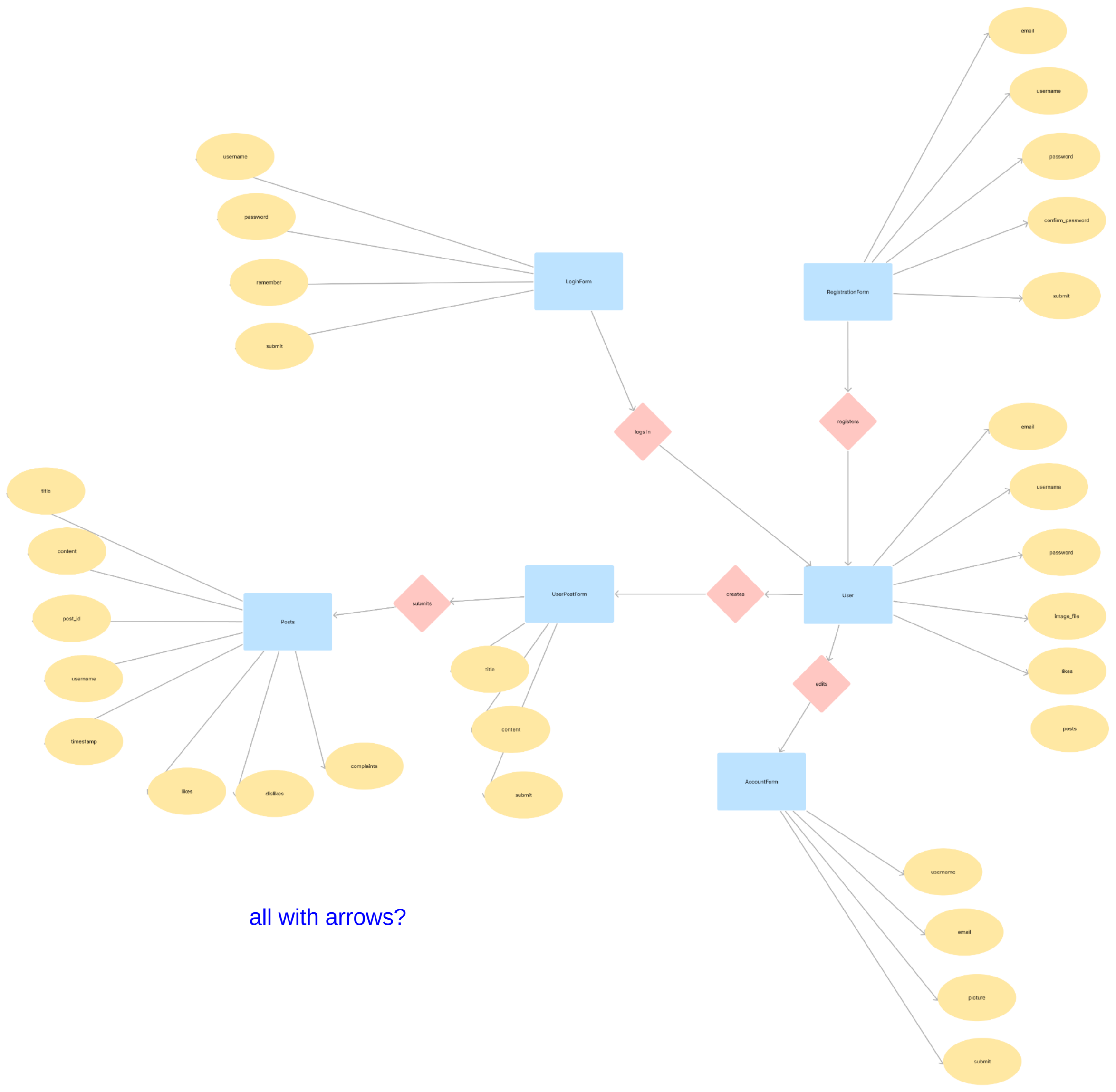
- Making posts:



- Customizing Profile:



3. E-R diagram for the entire system:



all with arrows?

In this E-R diagram, we show the relationships between our multiple classes, or entities, in our application. We use the register form in order to create an account, then use the login form in order to be logged in. If a user wants to create a new post, they must create a new post form in order for it to be submitted. A post carries for example, a timestamp, the author, content, and number of likes/dislikes. Users can also submit an edit account form where they can edit their profile picture, username, and/or email.

4. Detailed design:
   for EVERY method use pseudo-code to delineate the input/output and
   main functionalities

- The following pseudocode is for methods whose main functionalities have been implemented at this current time.
- **Chirphub.py:**

```
# necessary for flask_login to handle user authentication
user_loader(user_id):
  IF the user_id is an existing email:
      RETURN the User() obj that has that email
  # in the case that email is being changed or the user does not exist (DNE)
  else:
       user_id == user email
       IF no such email exists:
          RETURN None

# the following are nodes functions that describe the routes to each of out pages
home():
  Accesses user && posts dataFrames
  RETURN them as arguments to the home.html template

# while it has no input values, the route decorator has access to HTML GET and POST methods
register():
  IF the current_user is already logged in:
     Redirect to homepage
  IF the registration form returns no error:
     Create User(form.data)
     RETURN redirect to login-page

 RETURN the register.html template with register form passed as argument

# GET, POST passed to decorator
login():
   IF current_user is logged in:
     Redirect to homepage
   IF the login form returns no error:
      Check if the input username matches an user in the DataFrame
       IF the username match is found:
          Compare the username with the entered password
          IF username == password:
              Create instance = User(user.data)
              # use flask_login authentication function
```

```
            login_user(instance)
          ELSE redirect back to login-page
        ELSE redirect back to login-page
    RETURN the login.html template with login form passed in

# logout route relies on solely on flask_login method for implementation
logout():
  logout_user()
  RETURN redirect to homepage

# implementation for password reset not added
reset_request():
  RETURN reset_request.html

# function to add picture to sys DataFrame
picture_path(image_file) :
  img_path = combine("static/profile_pics", image_file filename)
  Save the image_file to the new path
  RETURN img_path

# GET, POST passed to the decorator
# login_required decorator used to make the page accessible only to logged-in users
account():
  IF account form returns no errors:
    # upon editing the file
    IF picture was entered into form:
      Add to DataFrame
      Set current_user's image_file to entered file

     # update DataFrame based on form data
     current_user username = form username
     current_user username = form username
    # functionality explained in data.py section
     updateUser(current_user)

    RETURN redirect to account-page

    # display current_user values before update
   ELIF request method is GET:
     set form.[username, email, && picture] to current_user values
    ELSE: print the form errors

    RETURN account.html with image_file and account form passed in

# The following functions are related to post creation in the sys

# ["GET", "POST"] are passed into the decorator
new_post():
   IF the post form returns no errors:
       Create a Post(form.data)
       RETURN redirect to homepage

  RETURN new_post.html with post-form passed in
single_post(post_id):
  # single_post.html renders the post indicated by the post_id
  RETURN single_post.html with current user && post DataFrames passed in

update_post(post_id):
  RETURN redirect for homepage

delete_post(post_id):
  RETURN redirect for homepage
```

- **Data.py:**

```
class User(UserMixin):
    # class variables
    users = pd.DataFrame()
    def __init__(self, email, username, password, image_file, likes, posts):
       set the initial values of the attributes

   load_users(cls):
       Set the users DataFrame = users in "data.csv" file

  createUser(cls, email, username, password,
image_file="static\profile_pics\default.png"")
     Load current users
     Create a new_user = User(email, username, password, image_file ,[],[])
     Create a DataFrame for the new_user
     Concatenate new_user to user DataFrame
     Write updated user DataFrame to "data.csv"
```

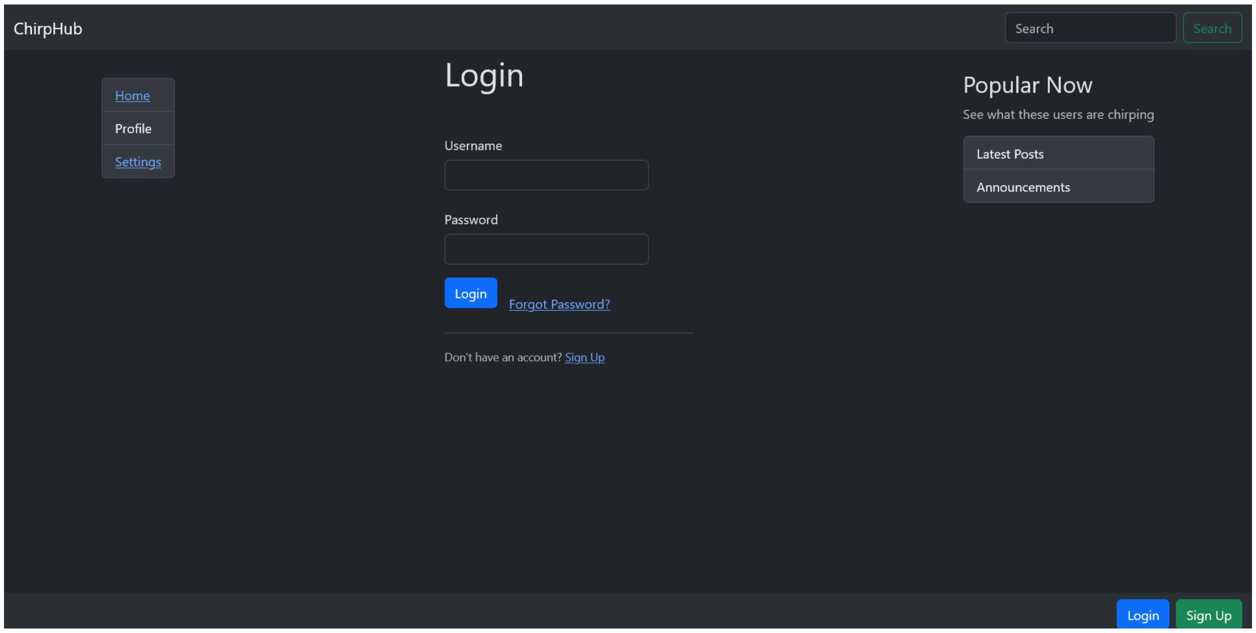
```
# where current_user is being edited
updateUser(cls, current_user):
  Load current users DataFrame
  IF the current_user's username can be found in DataFrame:
    user_index = the first matching username row
  ELIF the current_user's email can be found in DataFrame:
    user_index = the first matching email row

  IF user_index is not None:
    Update users DataFrame at user_index with current_user attributes
    Write updated DataFrame to "data.csv"
  ELSE: print ("Unable to update unknown user")

get_id(self):
  RETURN self.email

# the following are functions that override UserMixin methods
# necessary for user authentication by UserMixin
is_active(self):
  RETURN True
is_authenticated(self):
  RETURN True
is_anonymous(self):
  RETURN False
is_active(self):
  RETURN True
is_authenticated(self):
  RETURN True

 class Post:
   # class variables
   columns = ["title", "content", "username", "post_id", "date-posted"]
   posts = pd.DataFrame(columns=columns)
   post_id = 0

   def __init__(self, title, content, post_id, username):
        Initialize the attributes to the given values

  load_posts(cls):
```

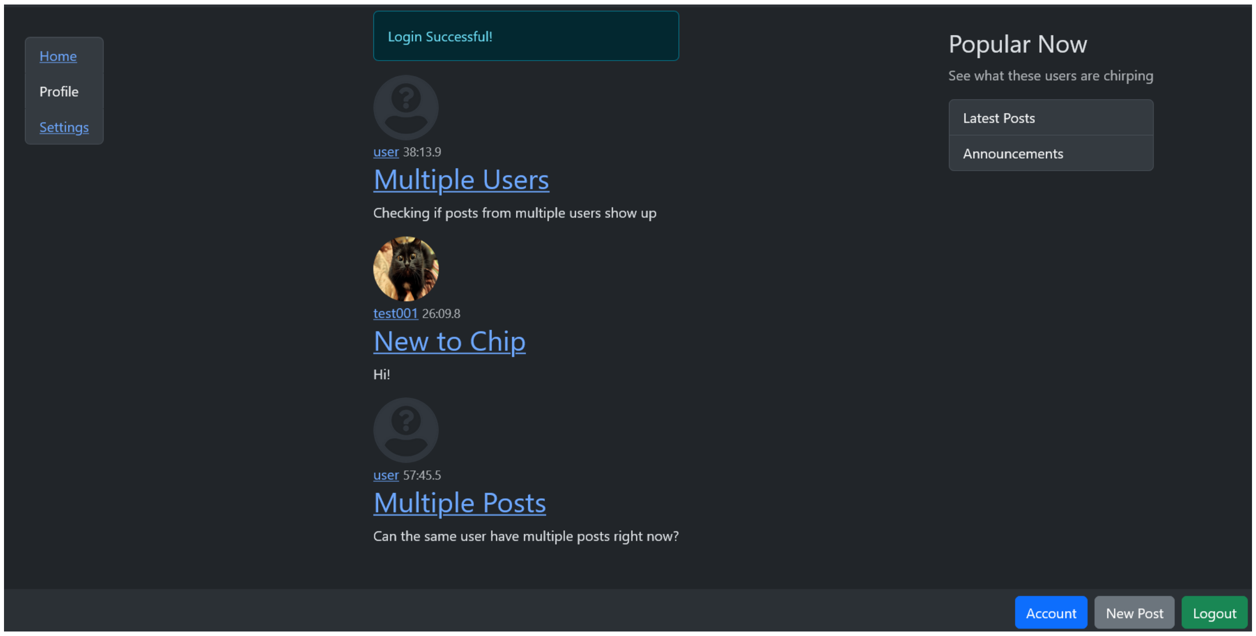
```
            Read the data from "posts.csv" to the post DataFrame

        createPost(cls, title, content, username, date_posted=datetime.utcnow()):
            Load posts
            new_post = new DataFrame([parameters])
            Concatenate new_post to cls.posts
            Update "post.csv"
            post_id += 1
```

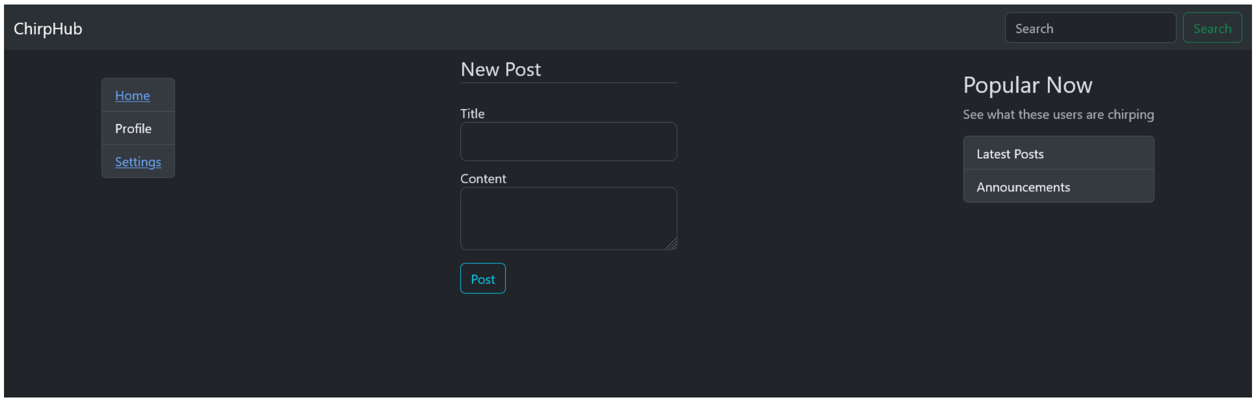
- Forms.py:

```
# The following classes create the forms returned by the routes in the main file
# each form is a class thanks to wtforms package
class LoginForm(FlaskForm):
  username = input required StringField
  password = input required PasswordField
  remember = BooleanField("Remember Me?")
  submit = SubmitField("Login")

class RegistrationForm(FlaskForm):
  email = input required StringField
  username = input required StringField
  password = input required PasswordField
  confirm_password = input required PasswordField, must match password
   submit = SubmitField

class EditAccountForm(FlaskForm):
  username = input required StringField
  email = input required StringField
  picture = FileField(), restricted to "jpg", "png" format
  submit = SubmitField

# used in editing and viewing single posts
class UserPostForm(FlaskForm):
  title = input required StringField()
  content = input required TextAreaField
  submit = SubmitField
```

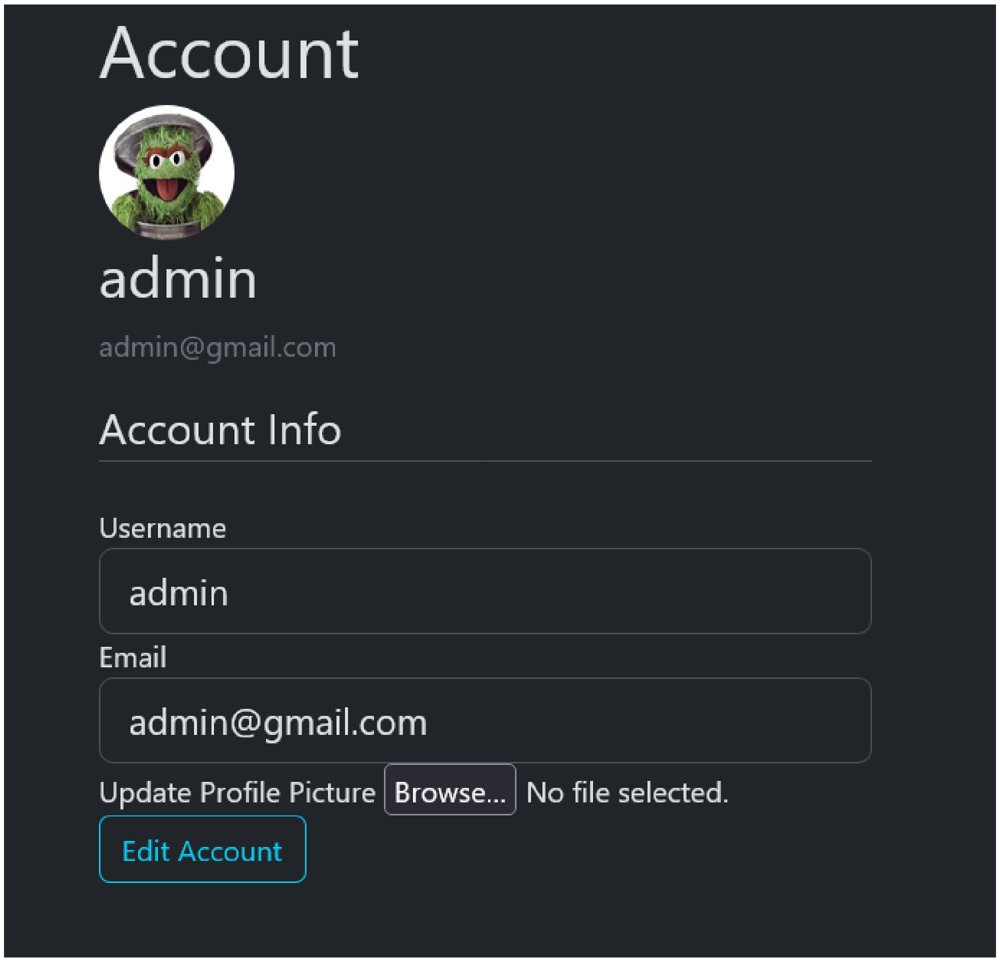
5. System screens:


ChirpHub
Search
Search
Home
Profile
Settings
Login
Username
Password
Login
Forgot Password?
Don't have an account? Sign Up
Popular Now
See what these users are chirping
Latest Posts
Announcements
Login
Sign Up


Home
Profile
Settings
Login Successful!
user 38:13.9
Multiple Users
Checking if posts from multiple users show up
test001 26:09.8
New to Chip
Hi!
user 57:45.5
Multiple Posts
Can the same user have multiple posts right now?
Popular Now
See what these users are chirping
Latest Posts
Announcements
Account
New Post
Logout


ChirpHub
Search
Search
Home
Profile
Settings
New Post
Title
Content
Post
Popular Now
See what these users are chirping
Latest Posts
Announcements

- Updating Account Info:


Account

admin

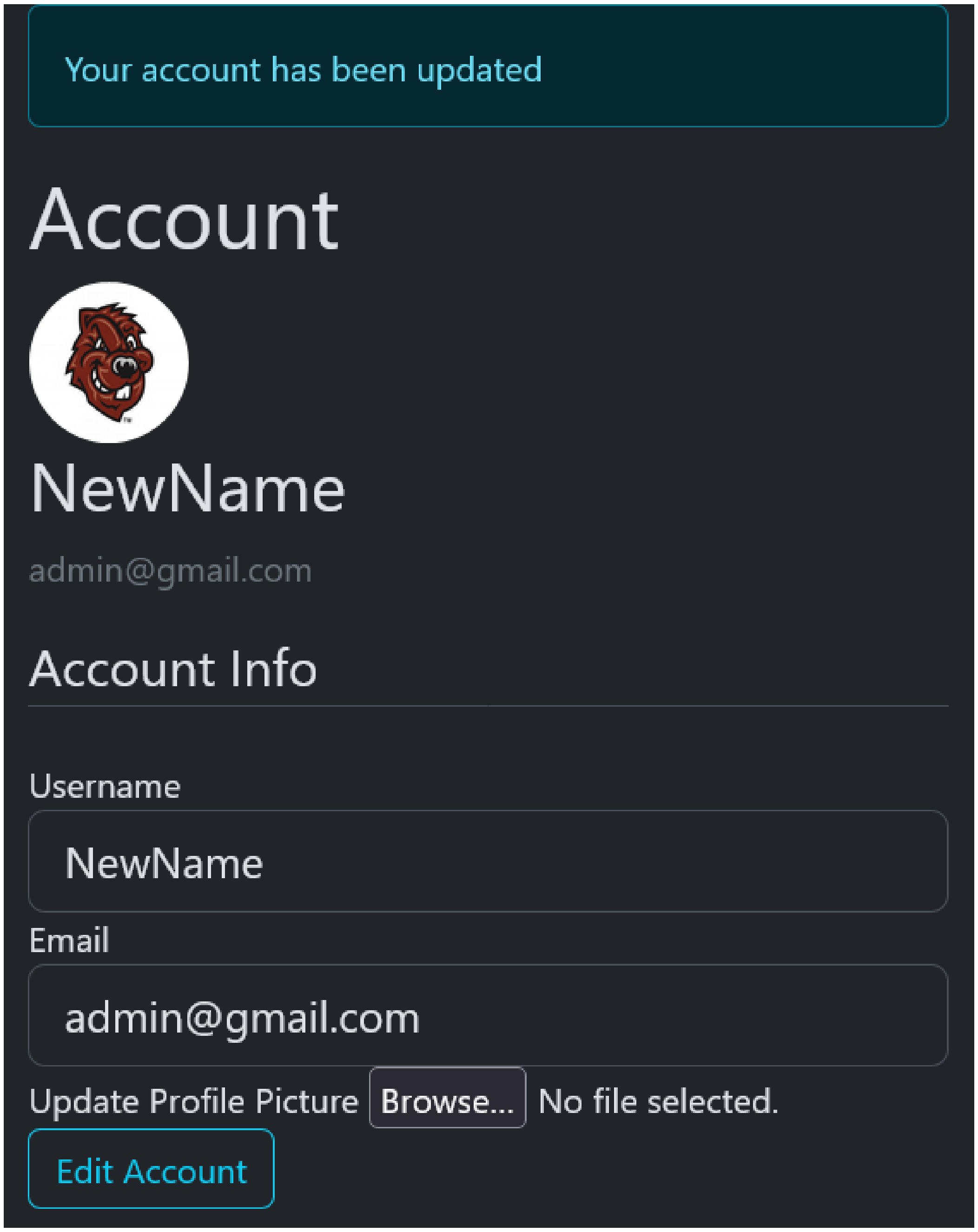
admin@gmail.com

Account Info

Username: admin

Email: admin@gmail.com

Update Profile Picture Browse... No file selected.



Your account has been updated

Account

NewName

admin@gmail.com

Account Info

Username

NewName

Email

admin@gmail.com

Update Profile Picture Browse... No file selected.

Edit Account

6. Memos of group meetings and possible concerns of team work:

Group members attended	**Meeting Dates (mm/dd/yyyy)**	**Plans discussed/notes**	**Tasks completed**	**To-do**	**Concerns**
Sajed Atwa, Joquanna Scott, Devin Munoz	09/14/2023	• Group formation • Thinking of ideas for project		• Choose framework(s) to work with. • Start making a plan for approach	
Sajed Atwa, Joquanna Scott, Devin Munoz	10/20/2023	• Git repo created	• Established git repo	• Begin plan for project structure	• Members not familiar with Python frameworks, Flask and Django
Sajed Atwa, Joquanna Scott, Devin Munoz	10/27/2023	• Finalizing report 1	• Report 1		
Joquanna Scott, Devin Munoz	10/28/2023	• New strategy due to smaller team		• Assign tasks to each member	• Large scope • Time constraint • Team size
Joquanna Scott, Devin Munoz	11/21/2023	• Finalizing report 2	• Report 2		

7. https://github.com/rockyrockz95/Mini-Twitter