# Team Rodda Reimer John

HW #09: Are You Devo

Jack Schluger, Bayle Smith-Salzberg, Haley Zeng, and Anya Keller

## **Program Components**

- Python files
  - o app.py all the app.routes
  - updateDB.py util file that writes to the database
  - readDB.py util file that reads from the database
  - adminAccount.py util file that controls user's account data (can send or update it)
- HTML
  - o login.html admin of blog logs in to be able to add posts
    - no register function: for the current version, there will only be one user (and thus one blog) with default credentials
    - default credentials can be changed by going to the /settings route
  - o settings.html for updating admin's information
  - o readPost.html (includes jinja loop)- for viewing posts
  - o writePost.html (includes HTML form)- for writing posts
- Database
  - posts.db stores the posts
    - each blog will have one table in the database
  - users.db stores user login information (to prepare for v2\_multiple\_users, we are using a database for this)
    - username
    - email
    - hashed password
    - display name

## **Component Relationships**

- app.py has the following routes:
  - 0 @app.route('/')
    - redirects to /readPost
  - o @app.route('/admin')
    - renders login.html containing login form
    - owner of a blog logs in to be able to edit their blog

- NO DIRECT LINK TO ADMIN LOGIN ON THE BLOG
- o @app.route('/authenticate')
  - gets form data from login.html form
  - uses adminAccount.py pull records from users.db and attempts to authenticate
  - if success, creates a session and redirects to /writePost
  - if failure, redirects back to /admin with a "Try Again" message
- o @app.route('/settings')
  - if blog owner is logged in, allows admin to edit their account information
    - changing password requires that user types in their current password
  - looks into users.db to get the user's current information and pre-fills it into form fields
- o @app.route('/updateAdmin')
  - gets data from settings form
  - uses util/adminAccount.py to get user's current data and update the changes
    - if password update requested, authenticate that Previous Password entered matches the record
  - returns back to /settings route with success/failure message, showing changes made
- o @app.route('/writePost')
  - if blog owner is logged in, renders writePost.html form
  - if not, redirects them to /admin
- o @app.route('/pushPost')
  - sends the writePost.html form data to utils/updateDB.py
- o @app.route('/readPost')
  - / route automatically goes here
  - gets data from utils/readDB.py
  - uses readPost.html to render feed of all the posts
- login.html
  - o has a form where blog owner logs in
  - o data is sent to /authenticate route
- settings.html
  - o has a form where blog owner can change their account information
  - o form fields preloaded with current information
- adminAccount.py
  - o main control for user's account information
  - o has functions that will
    - return the data on file
    - update the data on file
- writePost.html
  - o has a form which allows you to input a post

- two form components:
  - title
  - post contents
- The submit button sends the form data to /pushPost, which sends the data to utils/updateDB.py
- updateDB.py
  - uitl program
  - o accepts the form data from writePost.html and writes it to the database.
  - o stores title, post contents, date, time, and a unique post id
  - redirects to readPost route
- readPost.html renders the post data given by readDB.py using jinja
- readDB.py
  - Util program
  - o Reads from posts.db
  - Returns render\_template of readPost.html, passing the posts through a jinja loop

### **Database Schema**

#### posts.db

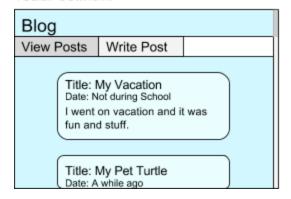
- in the interest of preparing for version 2:
  - each blog will have its own table of posts
  - o the table's name will correlate with the user's username
- each table will include columns for:
  - Title
  - Post Content
  - Date and Time
  - Post id

#### users.db

- one table containing all the users (to prepare for version 2)
- stores blog owner's account information
  - username
  - o email
  - hashed password
  - o display name

# Site Map for Front End

#### readPost.html



#### writePost.html



### **Tasks**

- Write app.py Jack
- Write login.html ~ Haley
- Write adminAccount.py ~ Haley
- Write settings.html ~ Haley
- Write writePost.html ~ Bayle
- Write /utils/updateDB.py ~ Anya
- Write /utils/readDB.py ~ Bayle
- Write readPost.html Jack

