P00: Move Slowly and Fix Things

Target ship date: {2022-11-15}
Mykolyk Fanclub (April Li, Shafiul Haque, David Deng)
SoftDev
P00: Move Slowly and Fix Things
2022-11-02
Time spent: 1.3 hours

Scenario Two: Your team has been contracted to create a web log hosting site.

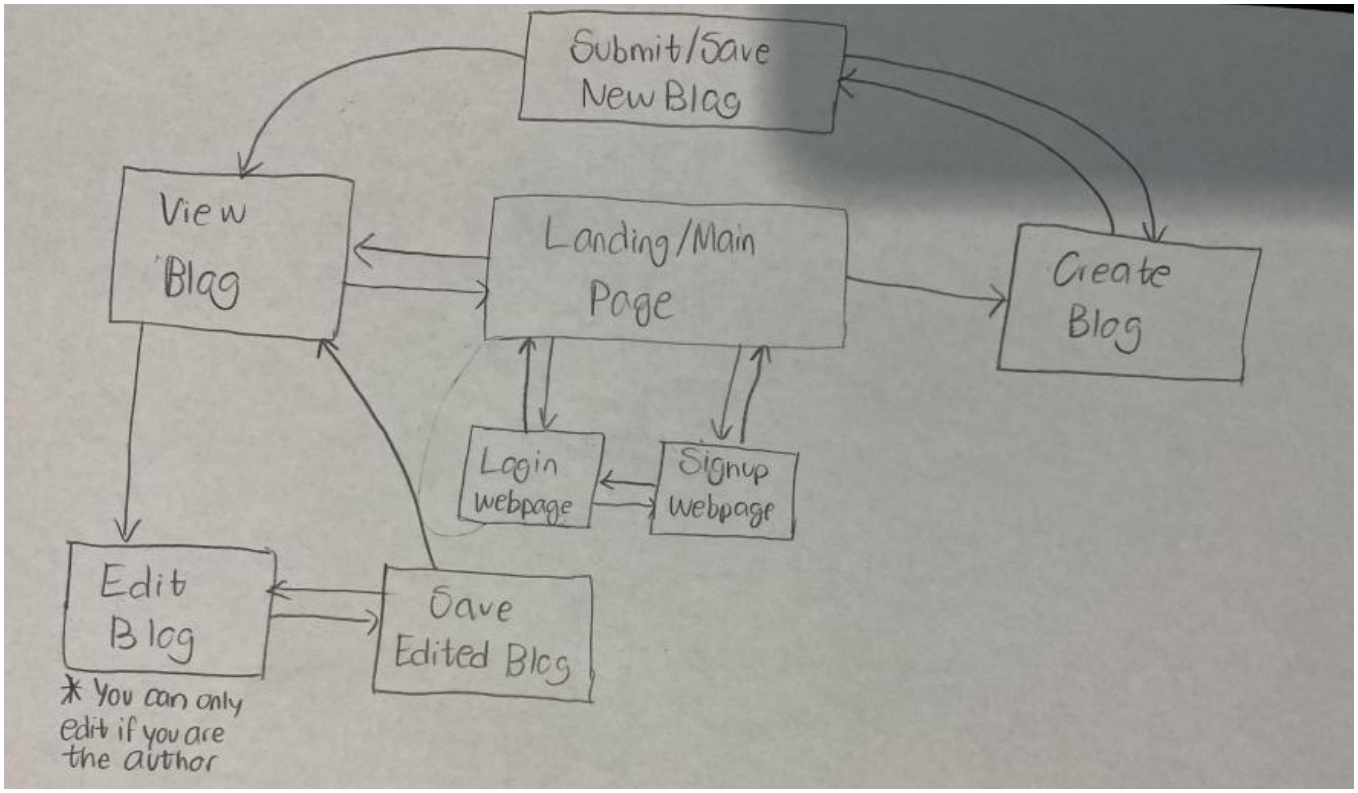
List of Program Components

- User accounts
    - Saving user information
    - User login and logout
- Blog manipulation
    - Create new blogs
    - Edit old blogs
- Html templates
    - Different templates for each page
    - Landing page which has a login button and signup button
    - Once the user is logged in, they are able to view all blogs
- Organizing data
    - Create tables with pre set headings
    - Obtain data from user input
    - Create tables using SQLite to store data
- NOTE: to save data for future uses, no drop table in the beginning of the code

Database Organization

- User information
    - Table to store user login information (username and password)
    - Teacher user will need a unique username
    - Data would be stored using SQLite
- Blog information
    - Table would be used to store all blogs
    - When the blog is edited, the old blog space will be replaced with the new one
    - When new blog is created, new blog space will appear
    - Each blog will be given a unique ID to help organize the data
    - The unique ID would be generated using the random() function
- Pages
    - Table would be used to store all pages of the website
    - Will store html template links
    - When data is loaded into the template links, the specific webpage would show

Site Map



Description of each path:

- main.html
    - The landing/main page
    - The path that flask would go through is /
    - We plan to have a design of our blog logo on the top of the page
    - [LOGIN]: this button on the main page would redirect to the login page
    - [SIGNUP]: this button on the main page would redirect to the signup page
    - If a user is logged in already, the main page is replaced with blog posts, which they are able to scroll through
- login.html
    - The login page
    - The path that flask would go through is /login
    - If a user is already logged in, then this page would automatically redirect to the main page
- signup.html
    - The signup page
    - The path that flask would go through is /signup
    - A user who isn't registered already would signup here
- blogpage.html
    - Viewing all the blogs
    - The path that flask would go through is /blogpage

    - This is the page that the main page would be redirected to once a user is logged in
    - This would be the same style as a platform like Piazza, where a user is able to scroll through all the blogs/posts, and then click one of their interest
    - If a specific blog post is clicked on, then the user would be redirected to viewblog[ID number] page
- createblog.html
    - Creating a new blog
    - The path that flask would go through is /createblog
    - A template would be displayed on the screen, where the user can choose a title for their blog and a space to write the actual components of the blog
- finalizeblog.html
    - Finalizing the creation of a new blog
    - The path that flask would go through is /finalizeblog
    - This would be the finalization screen, which would show a preview of how the actual blog would look like upon display
    - The user would have the option to either click Publish, which adds the blog to our database, or Discard, which removes the blog and all of its components entirely
- viewblog[ID number].html
    - Viewing the specific blog clicked on
    - The path that flask would go through is /viewblog
    - This would be used to view a specific blog page, which would request information from our database
- editblog[ID number].html
    - Editing the specific blog clicked on
    - The path that flask would go through is /editblog
    - Would be protected, since only the user who wrote the blog would be able to edit it
    - The same template would show up when you're trying to create a blog, but instead the information of the blog you are trying to edit would be on display
- saveblog[ID number].html
    - Editing the specific blog clicked on
    - The path that flask would go through is /saveblog
    - Would be protected, since only the user who wrote the blog would be able to edit it
    - This would be the finalization screen, which would show a preview of how the actual blog would look like upon display
    - The user would have the option to either click Publish, which adds the blog to our database, or Discard Changes, which reverts the blog to its original state

Description of each database

- app.py
    - Master python file
    - The main file that would be used to run the app, basically nothing works without it

    - Draws data from SQLite, and sends data to SQLite via Flask with all of the python files storing the data
- articles_db.py
    - A database of the articles
    - Would store information such as the date the article is published, who published the article, and the revision history
    - Draws data from articles.db
    - Sends data to the main python file
- users_db.py
    - A database of the users
    - Would store information such as the date the user registered, the articles that the user has written, and the users password (which would be encrypted)
    - Draws data from users.db
    - Sends data to the main python file
    - Once we get the main project out of the way, we could possibly pursue more features such as creating a user page where anyone can see which articles that user has written

Breakdown of tasks

- April
    - Data organization
    - Figuring out how to set up the SQLite
- Shafiul
    - Creating html templates
    - Specific to each website/webpage
    - Making sure the data fits in correctly with the templates
- David: app.py
    - Figuring out how login, logout
    - Linking and routing pages with each other