The Onions - Ishraq Mahid, Lucas Tom-Wong (LTW), Tomas Acuna
Softdev
P00 -- Design Dock
2021-10-27

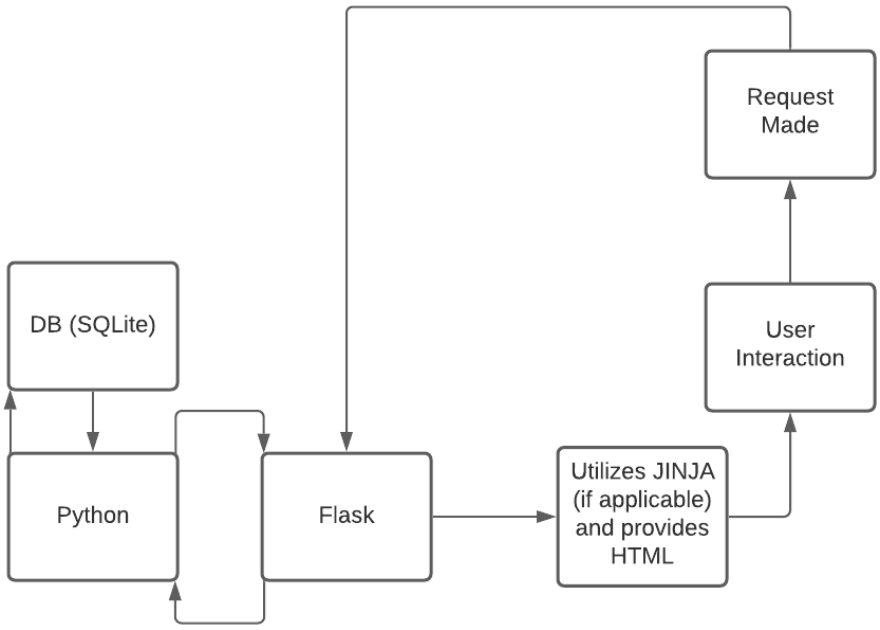
Program Components

Front end: Flask (html, Jinja 2); we will create the websites themselves, load them, and then deliver to the user utilizing Flask. Jinja2 and html templates and static pages will be useful for this as well, as Flask will deliver this material.

Database: SQLite (SQ but light) and CSV (separated values, but with commas); from the term “database,” it is implied that these tools will help us create a base for our data. SQLite will be our primary database, as implied by the assignment, and CSV will serve whatever function it may end up being useful for (although it is entirely possible we don’t use CSV at all).

Back end: snake (sssssssss); the python slithers in the background. It will allow us to do whatever data processing we may need in conjunction with Flask. We are literally coding in Python, and utilizing the different components within this language, connecting them to be one.

User Clicks on UI -> Flask Response -> Database Retrieval As Necessary -> danger noodle processing -> Flask -> Jinja -> HTML



Directory Path/Repository Structure

app/
    __init__.py
    static/
        css/
    templates/
Design.pdf
Design_wiki.pdf
Devlog.txt
flag.jpg
README.md
requirements.txt

Website Paths

"/":: login if not logged in; will be able to access site on login
    "/register" : register page
    "/<storyname>" : access desired story (functions differ based on user permissions)
    "/edit?name=<storyname>" : edit desired story (given user has permissions)

Home Page / Login Page
- When not logged in:
    - Asks for login with username and password
    - Has a register button which redirects to a register page
- When logged in:
    - Has a button for creating stories with a text field to put in its name
    - Shows a list of links to edited stories
    - Shows a list of links to unedited stories

Register Page
- Only appears when you press the register button from home/login page
- Asks for username and password for the website
    - Checks username among already registered usernames
        - If it already exists, error
        - Else, -> success

- Stores new username and password to database of usernames and passwords
- Then redirects you to the website

True Home Page (Logged In)
- Shows you a list of stories you have worked on
- Shows button to go to list of stores you can work on
- Maybe add a visual to show # of contributions?????

Story Pages:
- User perms depend on their contribution level to the story
- Text of the story is viewable depending on the user perms
- Will have edit option, which also depends on contribution level
- If they choose to edit (given they have permission) they will be redirected to the edit form

Every page should have a top bar, that leads to either logout/login or to the homepage, (this is where template would be handy).

Appearances, UI, and Frontend Funness (Flask and html and templates and static and other stuff) - with drawings to be added

---

Table Formatting

The **users** table holds the username and password of each user.

username TEXT	password TEXT
duckymcduck	quack quack
bobdylan	the moon is made of cheese
netlogoturtle	have a good day

The **stories** table holds the title of each story along with its most recent addition and everything written before that addition.

title TEXT	old_part TEXT	new_part TEXT
Lord of the Rings	frodo baggins liked submarines he sailed through the oceans	and then he sailed through all the streams
Doors of Stone		
Harry Potter		there once was a boy named harry

The **[username]** table exists for every user and holds the titles of the stories they've edited.

title TEXT
Lord of the Rings
Harry Potter

Other database stuff and important information that needs to be stored

- List of usernames in order to make sure no two users have same
- List of storyIDs to ensure no two have the same
- List of userIDs to ensure no two have the same. (We can probably remove userIDs/key numbers and just have id based on username)

Task Breakdown:

- Database Development & Maintenance [Tomas]
    - Importation of data
    - Methods to edit Database as desired
    - Methods to retrieve Database information as desired
    - SQLite
- User Interface and HTML Development & Maintenance [LTW & Ishraq]
    - Development of the UI

    - Creation of Templates and Statics
    - Website responsiveness
    - Flask, HTML, and JINJA
    - POST, GET, and Cookies
- Backend, Flask and Python Response and Data Manipulation (Danger noodle handling) [LTW & Ishraq & Tomas]
    - Taming the snek
    - THOUST APP SHALT NOT FAIL (Try Catch blocks and if statements)
    - Processing of Data, and overall linkages or misc jobs, related to the project
- PM [Ishraq]