Hot Cocoa - Hebe Huang, Josephine Lee, Annabel Zhang, Han Zhang
SoftDev
P00 -- Cafe of Stories | Design Doc
2021-11-11

Scenario One: Your team has been contracted to create a collaborative storytelling game/website.

- Users will have to register to use the site.
- Logged-in users can either start a new story or add to an existing story.
- When adding to a story,
  - Users are shown only the latest update to the story, not the whole thing.
  - A user can then add some amount of text to the story.
- Once a user has added to a story, they cannot add to it again.
- When creating a new story,
  - Users get to start with any amount of text and give the story a title.
  - Logged in users will be able to read any story they have contributed to on their homepage (the landing page after login).

<u>Templates</u>

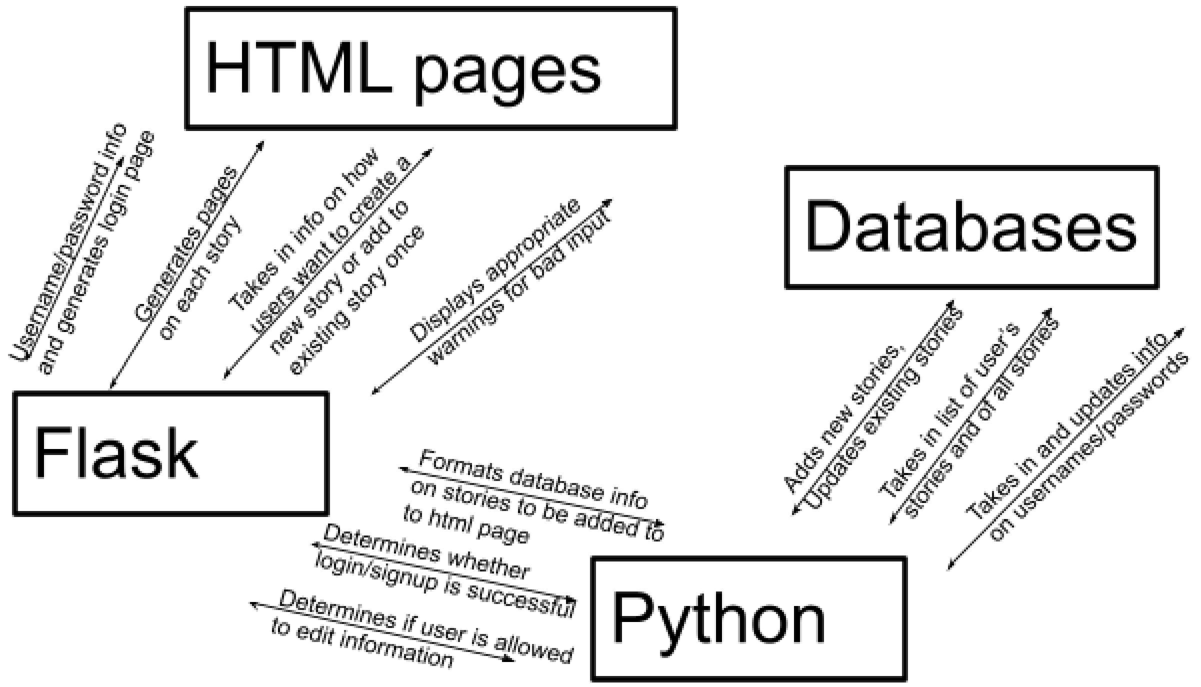
- login.html: Login page
- home.html: Home page
- story.html: Story viewing page
- createstory.html: Create new story page
- updatestory.html: Enter title of story to add to page
- addcontent.html: Adding content to existing story page

<u>Program Components:</u>

1. Python File
   a. Creates tables `users` and `stories`
   b. Verifies the account being logged in
   c. Uses `try except` blocks to reroute users if a url does not exist, otherwise sends them to the right page and template
   d. Inserts table values if a new account is created/a new story addition is created
2. HTML/Flask Templates
   a. `(@app.route("/"))` -> login page (login.html); contains login/signup input + login/signup failed screen

   b. `(@app.route("/auth"))` -> checks input from login.html with users database table; successful login leads to home page with an `Add to a Story` button, a `Create a New Story` button, and a `Logout` button as well as a list of the user's stories and input for the title of a story that the user wants to view
   c. `(@app.route("/signup"))` checks to see if input from login.html is valid; if so, adds the user to the database.
   d. `(@app.route("/logout"))` pops current user from session; leads back to login.html
   e. `(@app.route("/viewStory"))` -> links to inputted story, if inputted title from home.html is valid
   f. `(@app.route("/createstory"))` -> goes to createstory.html, a page with inputs for title and content of new story and home button
   g. `(@app.route("/uploadNewStory"))` -> adds input from createstory.html to database and returns to home.html
   h. `(@app.route("/addToStory"))` -> goes to page 1 of adding to a story (updatestory.html); has an input for the title of the story the user wishes to update, a list of all stories, and a home button
   i. `(@app.route("/addToTitle"))` -> determines whether title from updatestory.html is valid; if so goes to page 2 of adding to a story (addcontent.html). This displays the latest update of the requested story, an input field for the new content the user wishes to add, and a home button.
   j. `(@app.route("/uploadUpdatedStory"))` -> takes input from addcontent.html and adds it to the database then takes the user back to home.html
3. SQLite Database
   a. `stories` table (story title, story's latest update, full story, last user)
   b. `users` table (username, password)
   c. `[users]` table (titles of stories user contributed to)
4. CSS (To come after html is formatted)

Component Map:


HTML pages
Databases
Flask
Python
Username/password info and generates login page
Generates pages on each story
Takes in info on how users want to create a new story or add to existing story once
Displays appropriate warnings for bad input
Adds new stories, Updates existing stories
Takes in list of user's stories and of all stories
Takes in and updates info on usernames/passwords
Formats database info on stories to be added to to html page
Determines whether login/signup is successful
Determines if user is allowed to edit information

Database Organization:

Table of Users and Passwords

User	Password
<user1>	<password1>
<user2>	<password2>
...	...

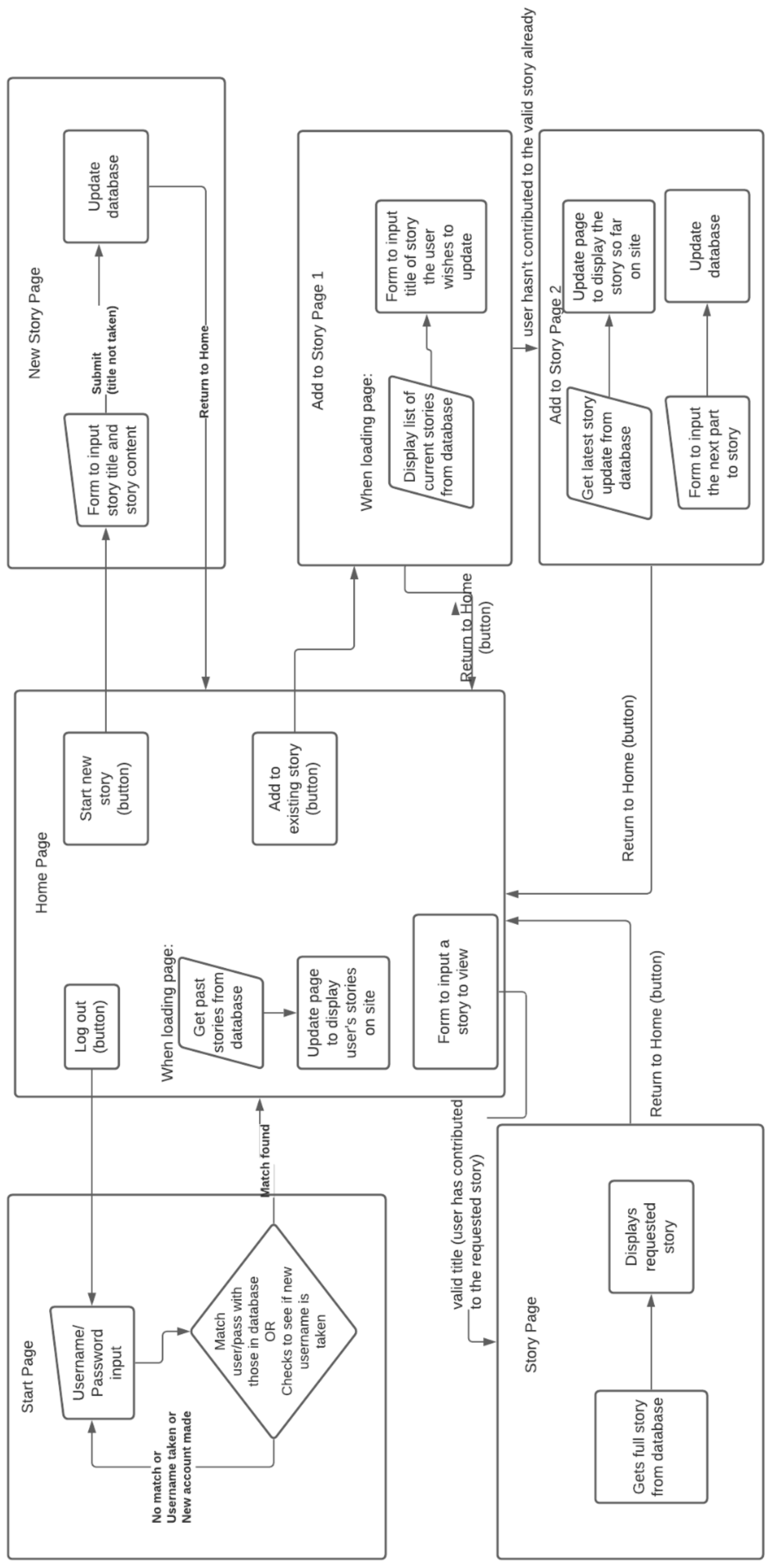
Table of Stories For Rendering Info

Story	Full Story	Story's Latest Update	Story's last editor
<Title of Story 1>	<Story 1 (full)>	<Story 1's Latest Update>	<Story 1's Last Editor>
<Title of Story 2>	<Story 2 (full)>	<Story 2's Latest Update>	<Story 2's Last Editor>
...	...	...	...

Table of Given User's Stories

User	**Story that user contributed to**
<username1>	<StoryName1>
<username2>	<StoryName2>
...	...

<u>Site Map</u>:



New Story Page
Form to input story title and story content
Submit (title not taken)
Update database
Return to Home
Add to Story Page 1
When loading page:
Display list of current stories from database
Form to input title of story the user wishes to update
user hasn't contributed to the valid story already
Add to Story Page 2
Get latest story update from database
Update page to display the story so far on site
Form to input the next part to story
Update database
Return to Home (button)
Return to Home (button)
Home Page
Start new story (button)
Add to existing story (button)
Log out (button)
When loading page:
Get past stories from database
Update page to display user's stories on site
Form to input a story to view
Return to Home (button)
valid title (user has contributed to the requested story)
Match found
Start Page
Username/ Password input
Match user/pass with those in database OR Checks to see if new username is taken
No match or Username taken or New account made
Story Page
Gets full story from database
Displays requested story

Tasks:

Josephine Lee: Project Manager; HTML templates, front end

Han Zhang: Python backend

Annabel Zhang: Python backend

Hebe Huang: Database generation