Team Green Monkeys; Daniel He, Faiyaz Rafee
SoftDev
P00-SCENARIO ONE
2022-11-15
time spent: 2 hrs

Target ship date: {2022-11-15}

Components of SCENARIO ONE:

_Users and login/logout functionality using SQLite

_Sign Up and Login pages:

- Usage of cookies to keep users logged in unless they log out.

_Displaying stories, adding on/ starting stories with the following features:

- Full stories can only be read after contributions
- Only last contribution displayed when adding on to a story
- Categorized into a predetermined set of genres (selected from a predetermined dropdown list using HTML select tag), alphabetical, etc
- Word limit when adding on, not applicable when creating a story
- Only contribute to a story once
- Must be logged in to access stories
- Title, body, authors displayed, date and time

_Homepage: body displays all stories the user has contributed to (display in a card form, clickable to view in its entirety)

_Explore Page: Displays all the stories the user hasn't contributed to. Shows the last contribution as well as author, genre, and date. Same card format as homepage stories. Users can click on a card to contribute to it.

_Contribute Page: Allows users to add body text with a limit of 1000 characters. Submitting it changes the latest contribution shown and also removes it from explore page and into the homepage. Also now allows users to read the entire story.

_CreateStory Page: Allows the user to create a story. The body has no word limit. Users must choose from the list of genres.

Software COMPONENTS to achieve:

_Flask web server serving different routes/ webpages

_Databases storing all stories and affiliated resources, users

_Python handling:

- Editing SQLite databases, (updating existing story databases)

_Links acting as buttons for navigation and activating methods

- Html and CSS for UI

_Forms for story writing (error handling for unfilled forms)
_Templates using jinja to insert dynamic, changing data into stagnant html, e.g. user contributed stories, available stories, etc

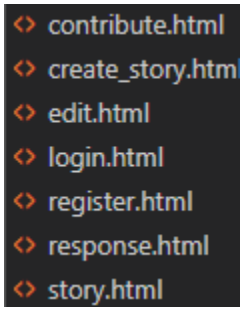
FLASK; work allotment: BOTH [flask], DANIEL [database], FAIYAZ [CSS]

__init__.py

Rudimentary **ROUTES** logics

- /homepage:
  - Accesses argument: username
  - Access table Users to specify selection in Stories table
  - Display all contributed stories in HomePage Template
- /stories/<type>/<title> and /find_story:
  - Takes arguments: title and type to render corresponding story
  - Type specifies whether the user is reading or contributing to a story, which corresponds to edit.html or story.html
- /creator and /edit_story:
  - Accesses arguments: username, story title, genre, body
  - Updates corresponding Stories table
- /login
  - Accesses arguments: username, password and checks with form inputs to log in user. Error message if wrong password/username
- /register
  - Accesses arguments: username, password and appends data to Users table
- /create_story and /contribute_story
  - renders template for create_story.html and contribute.html respectively. Contribute_story takes in argument: genre to render stories of a specific genre.
  - displays error messages

Templates:



Db.py

- **exodus()** - Opens/creates World.db. If tables exist, drop the tables and create new ones.

- **start()** - Opens/creates World.db, creates tables, and fills them with sample data
- **sample()** - Sample stories and sample users
- **user_exists(**a**)** - Determines if a user exists
- **title_exists(**a**)** - Determines if title exists
- **register_user(**username, password**)** - If user doesn't exist, registers user, returns true if successful
- **submit_story(**title, username, text, genre**)** - If title doesn't exist, submits story into table
- **login_user(**username, password**)** - returns true if password matches and user exists.
- **all_users()** - returns all users and stories
- **reset()** - reset database (runs exodus and sample)
- **contributed_stories(**username**)** - returns stories username has contributed to
- **non_contributed_stories(**username, genre**)** - returns stories username has not contributed to
- **non_contributed_strories_helper(**titles**)** - helper function
- **non_contributed_stories_titles(**username, genre**)** - returns titles of stories username hasn't contributed to
- **last_update**(title) - returns latest contribution of a particular title
- **full_story**(title) - returns every row in stories table with specified title ordered by date
- **eligible**(username, title) - returns true if user has contributed to the specified story
- **add_contribution**(title, username, body, genre) - add contribution to a story

DATABASE Tables:

"Users" (2 columns):

- Username TEXT P.K. | password TEXT not null

"Stories" (5 columns):

- Story Title TEXT not null, Author username TEXT not null, date TEXT not null, body of TEXT not null, genre TEXT not null

RELATIONSHIPPING:

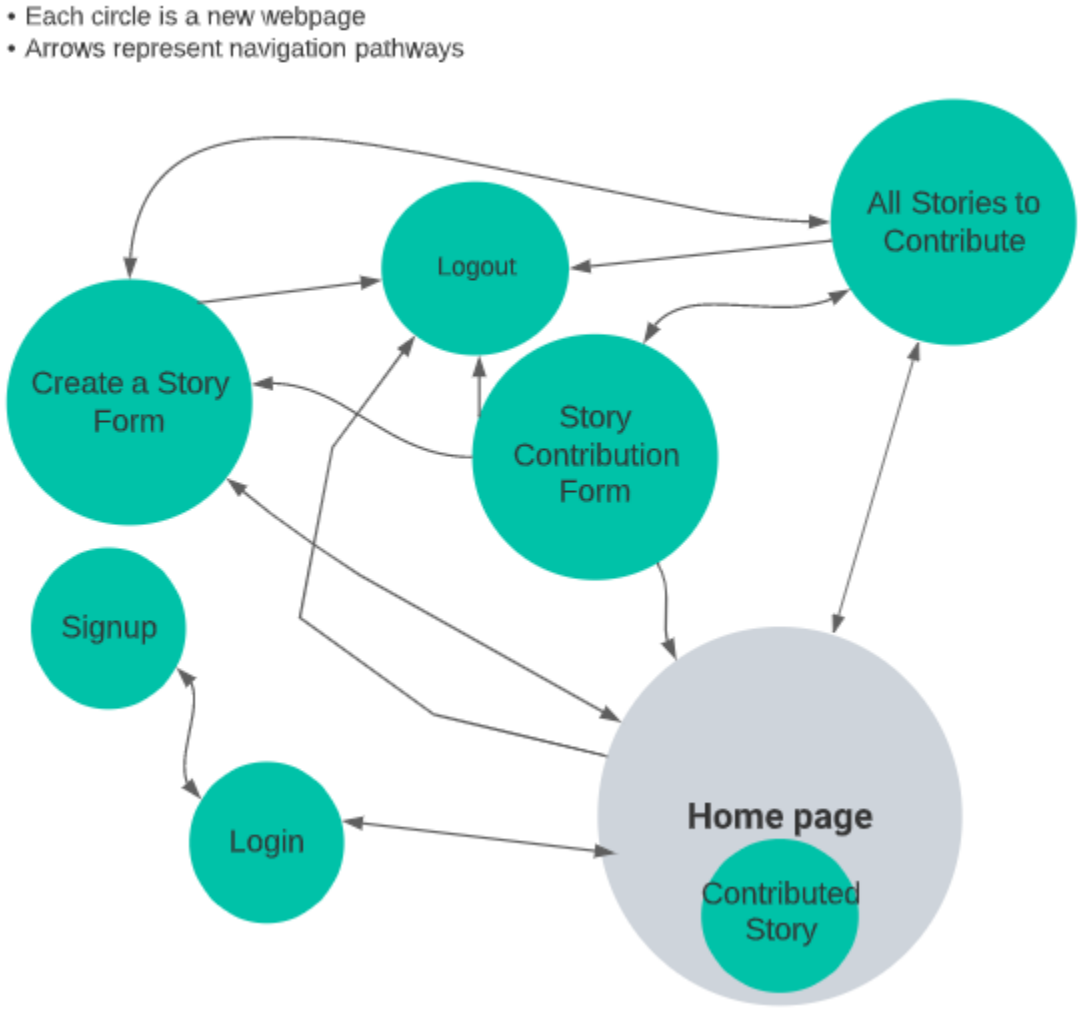
_ __init__.py will handle all the logic of navigation, as well as the logic of actions, such as editing or updating forms.

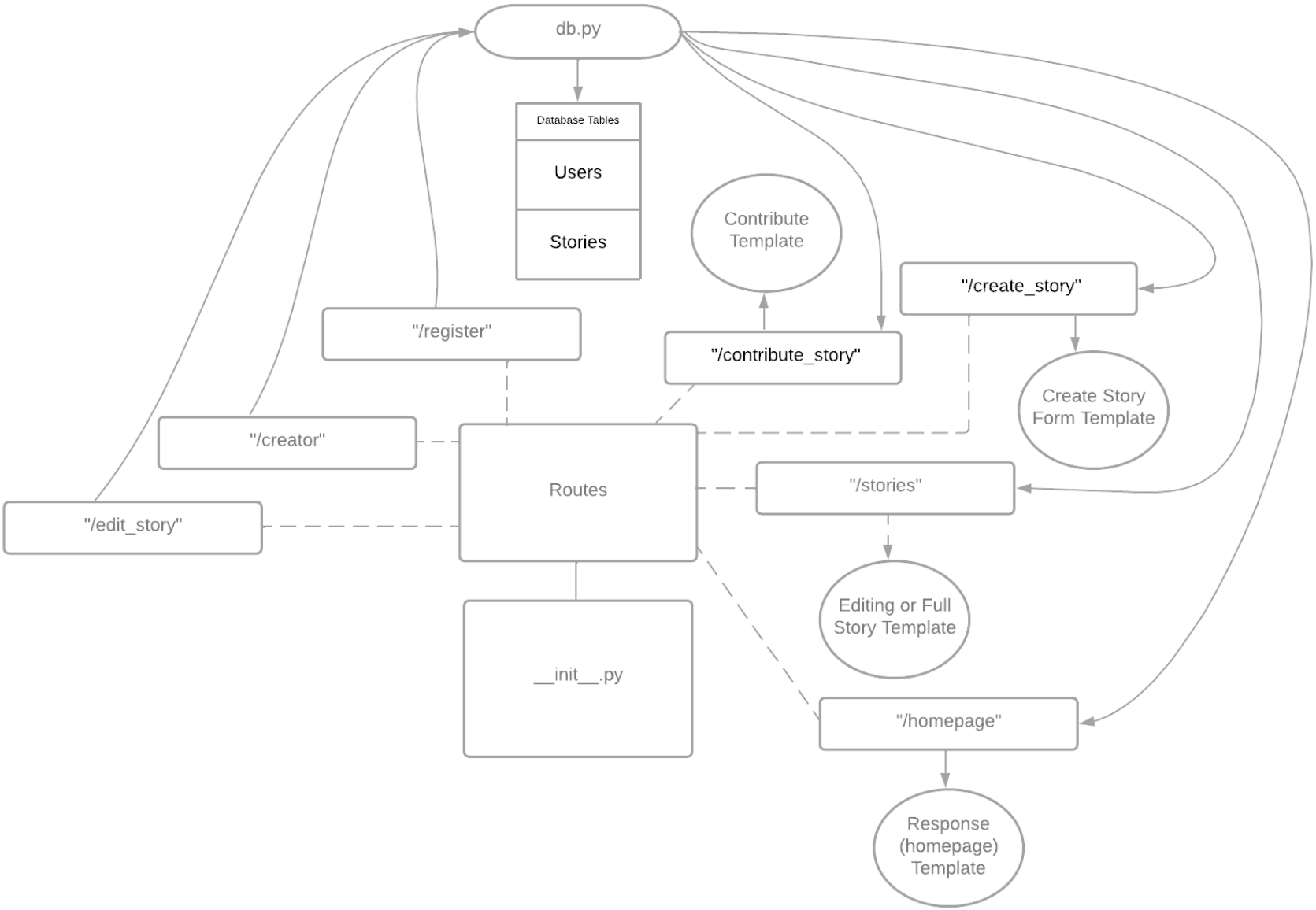
_Updating or editing forms will call upon methods of db.py

- submit_story(title, username, text, genre) adds a new story
- register_user(username, password) adds a new user
- add_contribution(title, username, body, genre) adds a contribution to an existing story

_When navigated to the correct routes, articles will be displayed by fetching data in the database (world.db) through methods in db.py which is called upon in corresponding app.routes in __init__.py

- contributed_stories(username) returns stories a user can read on their homepage
- non_contributed_stories(username, genre) and last_update(title) returns storie(s) a user can contribute to displayed in the contribute.html template along with their last update
- full_story(title) returns the full story a user can read




db.py
Database Tables
Users
Stories
Contribute Template
"/create_story"
"/register"
"/contribute_story"
Create Story Form Template
"/creator"
Routes
"/stories"
"/edit_story"
Editing or Full Story Template
__init__.py
"/homepage"
Response (homepage) Template