Pegasus Documentation

Release 0.1.0

Mariam Maarouf

Sep 27, 2019

Contents

CHAPTER 1

Introduction

1.1 About

Pegasus is a whiteboard collaboration app whose aim is to enable/facilitate realtime sharing of ideas in the form of components (for example: text, images, code, etc) that can be pieced together and edited by several people in a live session, and exported at any time.

At this point in time (*v0.1.0*), the only component available is text, along with the sidebar chat (which is available to everyone, edit privileges or not).

The plan is to add components, and features (like exporting the board as text instead of image), gradually, until the project reaches its initial purpose (stated above).

This project was part of the second edition of Learn IT, Girl, and would not have been possible without the continuous mentorship, support, and tolerance of 'newbie' questions courtesy of @daniel-j-h.

1.2 Rules

- Signed up users can create a board, to which they can give any number of other users access.
- **Each board can have 3 types of users with access:**
  - **Owner** Can do everything others can plus: edit board title, terminate board early, invite others and control their permissions, and delete the board.
  - **Editor** Can do everything viewers can plus: edit the components of the board.
  - **Viewer** Can only view the board, export its content at any time, and participate in the sidebar chat.
- **Editors** and **Viewers** can access the board either by signing into the website using the email they were invited with or accessing it through their unique invite link.
- Any signed in user with access to a certain board who is also not its **Owner** can remove themselves at any time.
- To avoid editing conflict, only one person can edit the board at a time.

Note: In order to simplify interaction with the server, most of the "heavy lifting" is done client-side. The server simply records and supplies data about the board as needed, and the client pieces it together in the DOM. The client-side board logic can be found on Github.

1.3 Installation

1. Clone the repo.

```
$ git clone https://github.com/mariamrf/pegasus.git
$ cd pegasus
```

2. Install virtualenv and activate it.

```
$ pip install virtualenv
$ virtualenv venv
$ . venv/bin/activate
$ pip install -r requirements.txt
```

3. Initialize the database.

```
$ chmod a+x init_db.py
$ ./init_db.py
```

4. Run the app.

```
$ chmod a+x run_pegasus.py
$ ./run_pegasus.py
```

Note: Default IP:port is 127.0.0.1:5000. You can change that by specifying the port and/or IP like this: `$ ./run_pegasus.py -ip IP_ADDRESS -port PORT_NUMBER`

CHAPTER 2

Docs

2.1 App

pegasus.**CSRF_ENABLED** = **True**
To be able to disable it when testing

pegasus.**before_request**()
Before database requests, connect, and turn on foreign keys.

pegasus.**connect_db**()
Connect to the SQLite database.

pegasus.**csrf_protect**()
Before database requests, in case of POST requests, check for the CSRF-protection token in the user's session. If present, pop (remove from session and get its value) then compare to the token submitted in the form. If there's no token or the token is not equal to the one from the form, abort the request.

pegasus.**init_db**()
Initialize the SQLite database. Used in init_db.py.

pegasus.**jsglue** = **<flask_jsglue.JSGlue object>**
JSGlue is what provides client-side js with Flask's url_for functionality without having to render the links via Jinja2

pegasus.**teardown_request**(*exception*)
If there's a database connection, close it.

2.2 Views

This module contains all the business logic related to the app 'views'/URLs and how the server interacts with the database. Because it's a simple enough app, for now, queries are directly written instead of being processed by something like SQLAlchemy. All AJAX POST functions return a new CSRF-protection token as the token is per request, not per session.

2.2.1 Division

1. Function definitions.
2. Basic/rendered views.
3. API for AJAX requests.

2.2.2 Database

In order for the following to make sense, here's the database schema/logic:

TABLE
USERS
BOARDS
BOARD_CONTENT
INVITES

pegasus.views.**change_password**()
Edit password for the logged in user. Front-end handles repeating the password twice before submitting.

pegasus.views.**create_board**()
Create a new board.

pegasus.views.**delete_board**(*boardID*)
Delete board. Only allows it in case the board in question

pegasus.views.**delete_component**(*boardID*, *componentID*)

Delete a component. Separated from editing for readability and possible future modification of the edit function. Also available to everyone with edit/owner access.

pegasus.views.**edit_board**(*boardID*)
Edit board title if the user is the owner.

pegasus.views.**edit_component**(*componentID*, *boardID*)
Edit a component's content (the text, etc). Available to anyone with edit/owner access to the board.

pegasus.views.**edit_invite**()
Edit the type of invitation/access to invited users. Available only to board owner.

pegasus.views.**edit_profile**()
Edit user info, such as username, email, and name for the logged in user.

pegasus.views.**generate_csrf_token**()
Create a CSRF-protection token if one doesn't already exist in the user's session and put it there.

pegasus.views.**get_components**(*boardID*)
Get all components of a board. This includes: - Chat, text, and other components along with all their relevant data (date, who, etc). - State of the board: locked/unlocked.

pegasus.views.**get_random_string**(*length=32*)
Generate a random string of length 32, used in `generate_csrf_token()`

pegasus.views.**get_user**(*userID*)
Get a user's username/name based on their ID. Available to everyone.

pegasus.views.**index**()
Helpers for the index page.

pegasus.views.**invite_user**(*email*, *boardID*)
Invite a user to a board via their email. Available only to the board owner.

pegasus.views.**invited_users**(*boardID*)
Get list of invited emails and the type of their invitations. Available to board owner only.

pegasus.views.**is_authorized**(*boardID*, *wantToEdit=False*)
Check if a certain signed in user (who by default doesn't want to edit the board) is authorized to access it, and if yes, what's the extent of their access? View/Edit/Owner. If edit/owner, can they edit now? (if the board is not locked, this function will lock it for them). Returns a hash that includes access (boolean), isOwner (boolean), canEditNow (boolean), and accessType (str)

pegasus.views.**is_owner**(*boardID*, *userID*)
Check if a user is the owner of a certain board.

pegasus.views.**lock_board**(*boardID*, *userID=None*, *userEmail=None*)
Called after making sure the board isn't currently locked and the user has editing access. Any sqlite3.Error s handled in calling function. Lock the board for 5 seconds.

pegasus.views.**login**()
Attempt login. If credentials check out, go to index. If not, render login template with error.

pegasus.views.**login_user**(*username*)
Login user using their username. Put username and userid (find in database) in their respective sessions.

pegasus.views.**logout**()
Logout currently logged in user and redirect to home.

pegasus.views.**mark_done**(*boardID*)
Mark a board as done before the 24 hours are up. Only available to the owner of the board.

pegasus.views.**post_components**(*boardID*)
Post a component to the board. Works for all types.

pegasus.views.**register_user**()
If not logged in and POST, register new user and go to index. If an error occurs, render the same register template again but with an error.

pegasus.views.**remove_self**()
Removes logged in user, who is not the owner of a board, from a certain board they'd been invited to.

pegasus.views.**show_board**(*boardID*)
Show board with a specified *boardID*.

Hierarchy of errors:

1. *404* : board not found.
2. *401* : not authorized. Person trying to view the board is not logged in with access to this board or does not have a (valid) invite code.

Renders the page (initially) according to the access type of the user (owner, edit, view).

pegasus.views.**show_profile**()
Render profile for logged in user.

pegasus.views.**validate_email**()
If email is available, return true. Else, if taken, return false. Used in registration.

pegasus.views.**validate_username**()
If username is available, return true. Else, if taken, return false. Used in registration.

2.3 Error handlers

How to handle common HTTP errors.

pegasus.errorhandlers.**bad_request**(*e*)
Render error template with the message: Bad Request and no details.

pegasus.errorhandlers.**forbidden**(*e*)
Render error template with the message: Forbidden and no details.

pegasus.errorhandlers.**gone**(*e*)
Render error template with the message: Page Gone and no details.

pegasus.errorhandlers.**internal_error**(*e*)
Render error template with the message: Internal Server Error and no details.

pegasus.errorhandlers.**not_found**(*e*)
Render error template with the message: Page Not Found and details.

pegasus.errorhandlers.**render_error**(*code*, *msg*, *det='Oops..'*)
Render template with variables appropriate to the error provided.

pegasus.errorhandlers.**unauthorized**(*e*)
Render error template with the message: Unauthorized and details.

2.4 Tests

This module contains all the tests related to the app.

p

t