CompSciCorner

Release 1

Austin LoPresto, Kai Hall, Brian Hinshaw

May 16, 2020

Table of Contents

1 CompSciCorner Main 1

2 CompSciCorner Events 3

3 CompSciCorner Trailers 5

4 CompSciCorner Users 7

Python Module Index 9

Index 11

CHAPTER 1

CompSciCorner Main

src.app.**admin** ( )
Returns function to render the admin login-page.

src.app.**admin_login** ( )
Renders login-page for admin account. Admins are able to access event/trailer submissions to approve them for display.

src.app.**approve_event** (*event_id*)
Approves an event and adds it to the active rotation

src.app.**approve_trailer** (*trailer_id*, *index=0*)
Approves a trailer from the trailer page

src.app.**choose_submission** ( )
Renders the submission page, where users can choose between either submitting a project or a trailer.

src.app.**confirm_submission** ( )
After user submits an event or trailer, they are redirected here and prompted to make another submission.

src.app.**deny_event** (*event_id*)
Rejects an event and removes it.

src.app.**deny_trailer** (*trailer_id*, *index=0*)
Denies a trailer from the trailer page

src.app.**display_page** ( )
Renders the main display that will be shown in the HNS 160's Hallway.

src.app.**handle_event_submission** ( )
Called after submit_event(), will take information user submitted and create an Event-object to be stored in MongoDB

src.app.**handle_trailer_submission** ( )
Called after submit_trailer(), will take information user submitted and create a Trailer-object to be stored in MongoDB

src.app.**home** ( )
Renders the submission choice page

src.app.**initialize_database**( )
Initializes MongoDB Database before application starts running

src.app.**load_user**(*id*)
loads a current user based on id

Param (str) id, corresponds to the id of a user in the database

src.app.**logout**( )
Redirects user to home and logs them out

src.app.**review_events**( )
Renders Event review page, User must be logged in as an admin to access.

src.app.**review_trailers**(*i*)
Renders the review page for trailers, User must be logged in as admin to access. The user can approve or deny pending submissions. Approved trailers join the queue to be put on the board, denied ones should send an email back to the user who submitted it with revisions they must apply.

src.app.**review_trailers_home**( )
Renders the homepage for admins reviewing trailers.

src.app.**submit_event**( )
Renders submission form for Events

src.app.**submit_project**( )
Renders submission form for Trailers

CompSciCorner Events

class src.models.event.**Event** (*author*, *email*, *title*, *date*, *time*, *location*, *description*, *_id=None*)

Events submitted by students to be displayed on the board

classmethod approve (*id*)

Takes in an id and removes it from pending_events, before adding it to approved_events :param id: (string)the id of the corresponding event in the database

date_time ( )

Returns (string) Converts date and time into a readable format to be displayed

classmethod deny (*id*)

Takes in an id and removes it from pending_events :param id: (string)the id of the corresponding event in the database

expire ( )

Remove self from database

classmethod from_mongo (*id*)

Returns (Event) Gets an event from the database using its id

classmethod get_approved ( )

Returns (list) returns a list of all approved events

classmethod get_pending ( )

Returns (list) returns a list of all pending events

json ( )

Method that converts the class into a json-format that can be stored in the database

Returns (dictionary) used for submitting an Event to the database

classmethod remove_expired ( )

Iterates through all approved events and removes those who have a date that has already passed

save_to_mongo ( )

Save self to database as a 'pending_event' to await for approval

CHAPTER 3

CompSciCorner Trailers

class src.models.trailer.**Trailer** (*author*, *email*, *display_email*, *title*, *trailer_path*, *date=datetime.datetime(2020, 5, 16, 2, 30, 13, 387341)*, *link=None*, *_id=None*)

Trailers submitted by students to be displayed on the board

classmethod approve (*id*)

Approves a trailer, moving it from one pending_trailers to queued_trailers

Parameters **id** – (string) id of the corresponding entry in the database

classmethod deny (*id*)

Denies a trailer, removing it from the pending_trailers

Parameters **id** – (string) id of the corresponding entry in the database

classmethod from_mongo (*id*)

Returns (Trailer) corresponding to the entry in the database with the same id

classmethod get_all (*collection*)

Get a list of all trailers in a specific collection

Parameters **collection** – (string) name of the collection we are looking at

Returns (list) consisting of Trailers from the database

json ( )

Method that converts the class into a json-format that can be stored in the database

Returns (dictionary) used for submitting Trailer to database

classmethod move (*entry*, *collection1*, *collection2*)

Moves an entry from one collection to another

Parameters

- **entry** – (Trailer) the item we want to move
- **collection1** – (string) collection where the entry currently exists
- **collection2** – (string) collection we want to move the entry to

classmethod remove_expired ( )

Iterates through all Trailers in the database, replacing those that have been in rotation for longer than 2 weeks with ones in the queue if they exist. Removed trailers are sent to the archived_-trailers collection in case someone wants to repurpose them in the future.

save_to_mongo (*collection='pending_trailers'*)
Save self to database as a 'pending_trailer' to await for approval

CHAPTER 4

CompSciCorner Users

class src.models.user.**User** (*username*, *password_hash*, *_id=None*)

User system being used to create admin accounts for approval process

static add_admin ( )

Creates an admin account with a default password and stores it in the database

check_password (*password*)

Takes in a password when trying to login, hashes it, and checks it matches the one in the database.

Parameters **password** – (string) password entered

Returns (boolean) whether the password matches the one in the database or not

get_id ( )

Returns the id associated with a user

Returns (string) user's id

classmethod get_user (*username*)

Gets a user from the database using their associated username

Parameters **username** – (string) the username of the user you're trying to retrieve

Returns (User) the corresponding user

classmethod get_user_by_id (*id*)

Gets a user from the database using their associated id

Parameters **id** – (string) the id of the user you're trying to retrieve

Returns (User) the corresponding user

json ( )

Method that converts the class into a json-format that can be stored in the database

Returns (dictionary) used for submitting User to database

save_to_mongo ( )

Inserts the user into the database

set_password (*new_password*)

Changes the password associated with the user in the database

Parameters **new_password** – (string) the new password

S

src
    src.app, 1
    src.models.event, 3
    src.models.trailer, 5
    src.models.user, 7

A

add_admin() (src.models.user.User static method), 7
admin() (in module src.app), 1
admin_login() (in module src.app), 1
approve() (src.models.event.Event class method), 3
approve() (src.models.trailer.Trailer class method), 5
approve_event() (in module src.app), 1
approve_trailer() (in module src.app), 1

C

check_password() (src.models.user.User method), 7
choose_submission() (in module src.app), 1
confirm_submission() (in module src.app), 1

D

date_time() (src.models.event.Event method), 3
deny() (src.models.event.Event class method), 3
deny() (src.models.trailer.Trailer class method), 5
deny_event() (in module src.app), 1
deny_trailer() (in module src.app), 1
display_page() (in module src.app), 1

E

Event (class in src.models.event), 3
expire() (src.models.event.Event method), 3

F

from_mongo() (src.models.event.Event class method), 3
from_mongo() (src.models.trailer.Trailer class method), 5

G

get_all() (src.models.trailer.Trailer class method), 5
get_approved() (src.models.event.Event class method), 3
get_id() (src.models.user.User method), 7
get_pending() (src.models.event.Event class method), 3
get_user() (src.models.user.User class method), 7
get_user_by_id() (src.models.user.User class method), 7

H

handle_event_submission() (in module src.app), 1
handle_trailer_submission() (in module src.app), 1
home() (in module src.app), 1

I

initialize_database() (in module src.app), 2

J

json() (src.models.event.Event method), 3
json() (src.models.trailer.Trailer method), 5
json() (src.models.user.User method), 7

L

load_user() (in module src.app), 2
logout() (in module src.app), 2

M

module
    src.app, 1
    src.models.event, 3

src.models.trailer, 5
src.models.user, 7
move() (src.models.trailer.Trailer class method), 5

R

remove_expired() (src.models.event.Event class method), 3
remove_expired() (src.models.trailer.Trailer class method), 5
review_events() (in module src.app), 2
review_trailers() (in module src.app), 2
review_trailers_home() (in module src.app), 2

S

save_to_mongo() (src.models.event.Event method), 3
save_to_mongo() (src.models.trailer.Trailer method), 6
save_to_mongo() (src.models.user.User method), 7
set_password() (src.models.user.User method), 7
src.app
module, 1
src.models.event
module, 3
src.models.trailer
module, 5
src.models.user
module, 7
submit_event() (in module src.app), 2
submit_project() (in module src.app), 2

T

Trailer (class in src.models.trailer), 5

U

User (class in src.models.user), 7