```python
from app import app, helper_functions, socketio, form_submissions, database_helpers
from app.models import *
from werkzeug.security import generate_password_hash, check_password_hash
from flask import redirect, render_template, request, session, url_for, abort, g, flash
from random import shuffle
from datetime import datetime
from sqlalchemy import and_, or_

CATEGORIES = ['All', 'Textbooks', 'Furniture', 'Food', 'Events', 'Software',
'Electronics',
 'Beauty and Personal Care', 'Clothes', 'School Supplies', 'Appliances']
CONTACT_METHOD = {
    "email": "Email (university provided)",
    "phonenumber": "Phone Number (if provided)",
    "personalemail": "Personal Email (if provided)"
}

# forces logout on browser close(); aka senses packets have stopped flowing
@socketio.on('disconnect')
def disconnect_user():
    logout()

######################################
# ROUTES START HERE
######################################
# Log the user out
@app.route('/logout')
def logout():
    session.pop('userid', None)
    return redirect(url_for('login'))

# Run at the beginning of each request before functions run to check if logged in
@app.before_request
def request_authentication():
    g.user = None
    if 'userid' in session:
        g.user = User.query.filter_by(userid=session['userid']).first()

### ERROR HANDLING PAGES
@app.route('/error')
def not_found_error_item():
    return render_template('404.html'), 404

@app.errorhandler(404)
def not_found_error(error):
    return render_template('404.html'), 404

@app.errorhandler(500)
def internal_error(error):
    db.session.rollback()
    return render_template('500.html'), 500

@app.route('/')
def slash_redirect():
    return redirect(url_for('login'))
######################### LOGIN ##############################################
# The login screen
@app.route('/login', methods=['GET', 'POST'])
def login(error=""):
    title = "Login to Craigversity!"
    LOGIN_ERROR = "Invalid information was submitted. Please try again!"
    error = ''
    if g.user:
        return redirect(url_for('user_home_screen'))
    if request.method == "POST":
        error = LOGIN_ERROR
        email = form_submissions.get_email(request.form['email'])
```

```
67          if not email == False:
68              user = User.query.filter_by(email=email).first()
69              if form_submissions.verify_password(user, request.form['password']):
70                  session['userid'] = user.userid
71                  return redirect(url_for('user_home_screen'))
72      return render_template('login.html', current_user_is_auth=False, error=error,
        page_title=title, css_file=helper_functions.generate_linked_files('login'))
73
74  ############################ ACCOUNT ROUTES ################################
75  # The create account screen
76  @app.route('/create-account', methods=['GET', 'POST'])
77  def create_account(error=""):
78      title = "Welcome to Craigversity!"
79      CREATE_ERROR = "Need to fill in ALL fields marked with an '*'"
80      errors = []
81      if g.user:
82          return redirect(url_for('user_home_screen'))
83      if request.method == 'POST':
84          [results, errors] = form_submissions.generate_new_account_form(request.form)
85          if len(errors) == 0:
86              added_successfully = database_helpers.add_user(results)
87              if added_successfully:
88                  return redirect(url_for('login'))
89              errors = "Could not process. Try again."
90      return render_template('create-account.html', current_user_is_auth=False,
        error=errors, page_title=title,
        css_file=helper_functions.generate_linked_files('create-account'), )
91
92  # The user screen
93  @app.route('/user', methods=['GET', 'POST'])
94  def users_account():
95      if g.user is None:
96          return redirect(url_for('login'))
97      account_info = g.user
98      title = "USER: " + g.user.username
99      if request.method == "GET":
100         userid = (request.args.get('userid'))
101         account_info = User.query.filter_by(userid=userid).first()
102         if account_info is None:
103             return redirect(url_for('not_found_error_item'))
104         postings    = database_helpers.get_postings(account_info.userid)
105         claims      = database_helpers.get_claims(account_info.userid)
106         sales       = database_helpers.get_sales(account_info.userid)
107         purchases   = database_helpers.get_purchases(account_info.userid)
108         title = "USER: " + account_info.username
109     return render_template('account-view.html', purchases=purchases, postings=postings,
        sales=sales, claims=claims, current_user_id=g.user.userid,
        current_user_is_auth=(g.user.userid > 0),  user_id=g.user.userid,
        CURRENT_USER_ID=g.user.userid, page_title=title,
        css_file=helper_functions.generate_linked_files('account-view'),
        account=account_info)
110
111 # The edit account screen
112 @app.route('/edit-account', methods=['GET', 'POST'])
113 def edit_account(error=""):
114     if g.user is None:
115         return redirect(url_for('login'))
116     title = 'Edit Account'
117     current_user = g.user
118     error = ''
119     if request.method == 'POST':
120         [result, error] = form_submissions.get_modified_account_info(request.form,
        g.user.userid)
121         if len(error) == 0 and check_password_hash(g.user.password,
        str(result['oldpassword'])):
122             if result['deleteaccount'] == "delete":
123                 database_helpers.delete_user(g.user.userid)
```

```
124                 if database_helpers.update_current_user(g.user, result):
125                     return redirect(url_for('login'))
126         return render_template('edit-account.html', current_user_id=g.user.userid,
            current_user_is_auth=(g.user.userid > 0),  error=error, current_user=current_user,
            CURRENT_USER_ID=g.user.userid, page_title=title,
            css_file=helper_functions.generate_linked_files('create-account'), )
127
128     ##################################################################
129     ###################### POSTINGS ##################################
130     # The new posting submission screen
131     @app.route('/new-posting-submission', methods=['GET', 'POST'])
132     def new_posting_submission(error=""):
133         title = "Submit a New Posting!"
134         error = []
135         if g.user is None:
136             return redirect(url_for('login'))
137         if request.method == 'POST':
138             [results, error] = form_submissions.get_form_create_post(request.form,
                CATEGORIES)
139             if len(error) == 0:
140                 database_helpers.add_new_post(results, g.user)
141                 return redirect(url_for('login'))
142         return render_template('create-posting-view.html', contact_options=CONTACT_METHOD,
            categories=CATEGORIES, current_user_id=g.user.userid, js_file="tag-javascript.js",
            current_user_is_auth=(g.user.userid > 0), page_title=title, error=error,
            css_file=helper_functions.generate_linked_files('create-posting-view'))
143
144     # NEED TO REWORK CONTACT METHOD!!!
145     @app.route('/edit-posting', methods=['GET', 'POST'])
146     def edit_posting(error=""):
147         title = 'Edit Posting'
148         if g.user is None:
149             return redirect(url_for('login'))
150
151         posting_info = database_helpers.get_post_id(request.args.get('postid'))
152         if posting_info is None or g.user.userid != posting_info.userid:
153             return redirect(url_for('user_home_screen'))
154
155         if request.method == 'POST':
156             [results, error] = form_submissions.get_form_create_post(request.form,
                CATEGORIES)
157             if len(error) == 0:
158                 print(results)
159                 database_helpers.modify_post_by_id(results, posting_info[0])
160                 return redirect(url_for('user_home_screen'))
161         return render_template('edit-posting-view.html',contact_options=CONTACT_METHOD,
            categories=CATEGORIES, js_file="tag-javascript.js", current_user_id=g.user.userid,
            current_user_is_auth=(g.user.userid > 0),  user_id=g.user.userid,
            CURRENT_USER_ID=g.user.userid, page_title=title, error=error,
            css_file=helper_functions.generate_linked_files('create-posting-view'),
            post=posting_info)
162
163     # The posting screen
164     @app.route('/posting', methods=['GET'])
165     def full_posting_view():
166         if g.user is None:
167             return redirect(url_for('login'))
168         #posting_info = {}
169         if request.method == 'GET':
170             [posting_info, user_info] =
                database_helpers.get_posting_by_id(request.args.get('postid'))
171             if posting_info is None:
172                 return redirect(url_for('not_found_error_item'))
173             title = "POSTING: " + posting_info.title
174         return render_template('full-posting-view.html',current_user_id=g.user.userid,
            current_user_is_auth=(g.user.userid > 0), page_title=title,
            css_file=helper_functions.generate_linked_files('full-posting-view'),
```

```
        post=posting_info, poster_info=user_info)

##########################################################################
# The home user logged in screen that lists postings
@app.route('/search-and-filter-postings', methods=['GET'])
def user_home_screen():
    title = "Search and Filter Postings!"
    page = 1
    if g.user is None:
        return redirect(url_for('login'))
    if request.method == 'GET':
        [submitted, randomize] = form_submissions.get_filters(request.args, True)
        categoryIsAll = True if submitted['category'] == 'All' else False
        if randomize:
            postings = database_helpers.generate_random_postings()
        else:
            if submitted['search'] == '':
                postings = Posting.query.filter(
                                    and_(
                                        Posting.price >= float(submitted['minPrice']),
                                        Posting.price <= float(submitted['maxPrice']),

                                         or_(Posting.category.contains(submitted['categor
                                         y']), categoryIsAll)
                                         )
                                    ).join(User).with_entities(
                                        Posting.postid, Posting.userid, User.username,
                                        Posting.title, Posting.price,
                                        Posting.description,
                                        User.rating)
            else:
                search_elems = form_submissions.get_search_elems(submitted['search'])
                postings = Posting.query.filter(and_(
                                        and_(Posting.tags.like(e) for e in search_elems),
                                        Posting.price >= float(submitted['minPrice']),
                                        Posting.price <= float(submitted['maxPrice']),

                                        or_(Posting.category.contains(submitted['category
                                        ']), categoryIsAll)
                                        )
                                    ).join(User).with_entities(
                                        Posting.postid, Posting.userid, User.username,
                                        Posting.title, Posting.price,
                                        Posting.description,
                                        User.rating)
    else:
        submitted = form_submissions.get_filters('', True)
        postings = database_helpers.generate_random_postings()

    page = form_submissions.get_page_number(submitted['page'], postings.count(),
    app.config['POSTS_PER_PAGE'])
    postings = postings.paginate(page, app.config['POSTS_PER_PAGE'], False)
    next_url = url_for('user_home_screen',search=submitted['search'],
    category=submitted['category'], minPrice=submitted['minPrice'],
    maxPrice=submitted['maxPrice'], page=postings.next_num) if postings.has_next else
    None
    prev_url = url_for('user_home_screen',search=submitted['search'],
    category=submitted['category'], minPrice=submitted['minPrice'],
    maxPrice=submitted['maxPrice'], page=postings.prev_num) if postings.has_prev else
    None
    return render_template('user-view.html', next_url=next_url, prev_url=prev_url,
    page=page, categories=CATEGORIES, able_to_filter=True, submitted=submitted,
    current_user_id=g.user.userid, current_user_is_auth=(g.user.userid > 0),
    page_title=title, css_file=helper_functions.generate_linked_files('user-view'),
    filtered_postings=postings.items)

########################### CLAIMS ########################################
```

```python
# the claim pages
@app.route('/claim', methods=['GET', 'POST'])
def claim_submission():
    if g.user is None:
        return redirect(url_for('login'))
    title = "Claim Submission"
    error = ''
    IsSeller = False
    post_info = {'title': '', 'postid': '', 'username': '' }
    try:
        [posting_info, poster_info] =
        database_helpers.get_posting_by_id(request.args.get('postid'))
        if posting_info is None:
            return redirect(url_for('error'))
        post_info = helper_functions.get_post_info_claims(posting_info, poster_info)
        isSeller = (posting_info.userid == g.user.userid)
        if isSeller:
            title = "Seller " + title
        else:
            title = "Buyer " + title
    except Exception as e:
        error = "Please Try Resubmitting. Something went Wrong."
        return render_template('claim.html', error=error, post=post_info,
        isSeller=False, current_user_id=g.user.userid,
        current_user_is_auth=(g.user.userid > 0), page_title=title,
        css_file=helper_functions.generate_linked_files('claim') )

    if request.method == 'POST':
        [submitted, error] = form_submissions.get_new_claims_form(request.form, g.user,
        post_info['postid'])
        if len(error) == 0:
            [completed_claim, claim] = database_helpers.add_claim(submitted,
            post_info['postid'], g.user)
            if completed_claim:
                is_transaction_completed = database_helpers.check_for_transaction(claim)
                if is_transaction_completed:
                    db.session.commit()
                    return redirect(url_for('claim_completion',
                    is_transaction_complete=True ))
                elif is_transaction_completed is not None:
                    db.session.commit()
                    return redirect(url_for('claim_completion',
                    is_transaction_complete=False ))
            error = "Please resubmit your claim. There was an issue. You may have
            entered invalid data."
    return render_template('claim.html', error=error, post=post_info,
    isSeller=isSeller, current_user_id=g.user.userid,
    current_user_is_auth=(g.user.userid > 0), page_title=title,
    css_file=helper_functions.generate_linked_files('claim') )

@app.route('/claim-complete', methods=['GET', 'POST'])
def claim_completion(is_transaction_complete=False):
    if g.user is None:
        return redirect(url_for('login'))
    is_transaction_complete = request.args.get('is_transaction_complete')
    print(is_transaction_complete)
    return render_template('claim-complete.html',
    is_transaction_complete=is_transaction_complete=="True", title="Claim Complete",
    error='', current_user_id=g.user.userid, current_user_is_auth=(g.user.userid > 0),
    css_file=helper_functions.generate_linked_files('claim'))

# The HELP page
@app.route('/help-and-FAQ')
def help():
    return render_template('help.html')


```

```
276 @app.route('/remove-posting', methods=['GET', 'POST'])
277 def remove_posting_view(error=""):
278     title = 'Remove Posting'
279     if g.user is None:
280         return redirect(url_for('login'))
281 
282     posting_info = database_helpers.get_post_id(request.args.get('postid'))
283     if posting_info is None or g.user.userid != posting_info.userid:
284         return redirect(url_for('user_home_screen'))
285 
286     if request.method == 'POST':
287         posting_info_remove =
        Posting.query.filter_by(postid=request.args.get('postid')).first()
288         db.session.delete(posting_info_remove)
289         db.session.commit()
290         return redirect(url_for('user_home_screen'))
291 
292     return render_template('remove-posting-view.html',contact_options=CONTACT_METHOD,
        categories=CATEGORIES, js_file="tag-javascript.js", current_user_id=g.user.userid,
        current_user_is_auth=(g.user.userid > 0),  user_id=g.user.userid,
        CURRENT_USER_ID=g.user.userid, page_title=title, error=error,
        css_file=helper_functions.generate_linked_files('create-posting-view'),
        post=posting_info)
293 
294 # Admin view of users
295 @app.route('/admin-view/users', methods=['GET', 'POST'])
296 def admin_view_users():
297     if g.user is None:
298         return redirect(url_for('login'))
299     elif g.user.userid != 1:
300         return redirect(url_for('user_home_screen'))
301     else:
302         if request.method == 'POST':
303             database_helpers.delete_user(request.form.get('user_id'))
304         UserQuery = User.query.order_by(User.userid).all()
305         return render_template('admin-view-users.html', UserQuery=UserQuery)
306 
307 # Admin view of postings
308 @app.route('/admin-view/postings', methods=['GET', 'POST'])
309 def admin_view_postings():
310     if g.user is None:
311         return redirect(url_for('login'))
312     elif g.user.userid != 1:
313         return redirect(url_for('user_home_screen'))
314     else:
315         if request.method == 'POST':
316             database_helpers.archivedPost(request.form.get('post_id'))
317             db.session.commit()
318         PostingQuery = Posting.query.order_by(Posting.postid).all()
319         return render_template('admin-view-postings.html', PostingQuery=PostingQuery)
320 
321 # Admin view of creating accounts
322 @app.route('/admin-view/create-account', methods=['GET', 'POST'])
323 def admin_view_create_account(error=""):
324     if g.user is None:
325         return redirect(url_for('login'))
326     elif g.user.userid != 1:
327         return redirect(url_for('user_home_screen'))
328     else:
329         title = "Create a User"
330         CREATE_ERROR = "Need to fill in ALL fields marked with an '*'"
331         errors = []
332         if request.method == 'POST':
333             [results, errors] = form_submissions.generate_new_account_form(request.form)
334             if len(errors) == 0:
335                 added_successfully = database_helpers.add_user(results)
336                 if added_successfully:
```

```
337                     return redirect(url_for('admin_view_users'))
338                 errors = "Could not process. Try again."
339         return render_template('create-account.html', current_user_is_auth=False,
            error=errors, page_title=title,
            css_file=helper_functions.generate_linked_files('create-account'), )
340
```