```
import sys
sys.path.append("..")
from unittest import TestCase
from app import app
from models import db, User, Story, Query
from forms import RegisterForm, LoginForm, SearchForm
from sqlalchemy import exc
from psycopg2.errors import UniqueViolation


app.config['SQLALCHEMY_DATABASE_URI'] = 'postgresql:///newstracker-test'
app.config['SQLALCHEMY_ECHO'] = False
app.config['TESTING'] = True
app.config['DEBUG_TB_HOSTS'] = ['dont-show-debug-toolbar']
app.config['WTF_CSRF_ENABLED'] = False
db.drop_all()
db.create_all()


registration_data = {
'form-register-username':'user1',
'form-register-password':'pass1',
'form-register-email':'test@email.com',
'form-register-first_name':'test',
'form-register-last_name':'user'
}

from flask import session
from flask_session import Session
app.config['SESSION_TYPE'] = 'redis'
app.config['SESSION_USE_SIGNER'] = True
```

```
app.config['SESSION_PERMANENT'] = False
CURR_USER_KEY = "curr_user"
server_session = Session(app)

class No_User(TestCase):

    def setUp(self):
        """Add sample user."""

        User.query.delete()

        user = User.register(
            "testuser", "test4444", "test@test.com", "test", "user")
        db.session.add(user)
        db.session.commit()
        self.user_id = user.id

    def tearDown(self):
        db.session.rollback()

    def test_forbidden_routes(self):
        """Tests Redirects"""
        with app.test_client() as client:
            res = client.get("/search")
            self.assertEqual(res.status_code, 302)
            self.assertEqual(res.location, "http://localhost/login")

            res2 = client.get("/search/results")
            self.assertEqual(res2.status_code, 302)
            self.assertEqual(res2.location, "http://localhost/login")
```

```
        res3 = client.get("/user/saved")
        self.assertEqual(res3.status_code, 302)
        self.assertEqual(res3.location, "http://localhost/login")

        res4 = client.get("/logout")
        self.assertEqual(res4.status_code, 302)
        self.assertEqual(res4.location, "http://localhost/login")


def test_redirection_followed(self):
    """Test that redirects are followed"""
    with app.test_client() as client:
        res = client.get("/search", follow_redirects=True)
        self.assertNotEqual(res.status_code, 302)
        html = res.get_data(as_text=True)
        self.assertIn('You do not have permission', html)
        res2 = client.get("/search/results", follow_redirects=True)
        self.assertNotEqual(res2.status_code, 302)
        html2 = res2.get_data(as_text=True)
        self.assertIn('You do not have permission', html2)
        res3 = client.get("user/saved", follow_redirects=True)
        self.assertNotEqual(res3.status_code, 302)
        html3 = res3.get_data(as_text=True)
        self.assertIn('You do not have permission', html3)


def test_404(self):
    with app.test_client() as client:
        resp = client.get("/this/route/does/not/exist")
        self.assertEqual(resp.status_code, 404)
```

```
    def test_get_login(self):
        """Simulates GET Request to check for status code and HTML"""
        with app.test_client() as client:
            res = client.get("/login")
            html = res.get_data(as_text=True)
            self.assertEqual(res.status_code, 200)
            self.assertIn('<h1 class="form display-3 text-center">Login</h1>', html)


    def test_get_register(self):
        """Simulates GET Request to check for status code and HTML"""
        with app.test_client() as client:
            res = client.get("/register")
            html = res.get_data(as_text=True)
            self.assertEqual(res.status_code, 200)
            self.assertIn('<h1 class="display-3 text-center">Register</h1>', html)

    def test_simple_search(self):
        """Simulates simple search"""
        with app.test_client() as client:
            #tests results when expected
            res = client.get("/search/simple?search=china")
            html = res.get_data(as_text=True)
            self.assertEqual(res.status_code, 200)
            self.assertNotIn("We weren't able to find anything. Try checking your spelling or widening
your search query.", html)

            #tests no results when expected
            res2 = client.get("/search/simple?search=12345567890qwertyuio")
            html2 = res2.get_data(as_text=True)
```

```
        self.assertIn("We weren't able to find anything. Try checking your spelling or widening your search query.", html2)


    def test_headlines(self):
        """Simulates non-user selecting headlines tab"""
        with app.test_client() as client:
            res = client.get("/headlines")
            self.assertEqual(res.status_code, 200)
            html = res.get_data(as_text=True)
            #tests button values on stories
            self.assertIn("Get Polarity", html)
            self.assertIn("Get Subjectivity", html)
            self.assertIn("Save Story", html)
            #tests html elements and classes
            self.assertIn('<div class="container">', html)
            self.assertIn('<section id="stories">', html)


    # def test_home_page(self):
    #     """Simulates non-user selecting headlines tab"""
    #     with app.test_client() as client:
    #         res = client.get("/")
    #         self.assertEqual(res.status_code, 200)
    #         html = res.get_data(as_text=True)
    #         #tests all categories are appearing
    #         self.assertIn("Business", html)
    #         self.assertIn("Science", html)
    #         self.assertIn("Health", html)
    #         self.assertIn("Entertainment", html)
    #         self.assertIn("Sports", html)
    #         self.assertIn("Technology", html)
```

```
#       #tests html elements and classes
#       self.assertIn('What can I do with NewsTracker?</h3>', html)
#       self.assertIn('Non-Users</h4>', html)
#       self.assertIn('Get Headlines</h5>', html)
#       self.assertIn('Search by Keyword</h5>', html)
#       self.assertIn('Browse by Category</h5>', html)
#       self.assertIn('Users</h4>', html)
#       self.assertIn('Detailed Search</h5>', html)
#       self.assertIn('Sentiment Analysis Tools</h5>', html)
#       self.assertIn('Saved Stories</h5>', html)
#       self.assertIn('Upcoming Features</h4>', html)
#       self.assertIn('Amplified User Features</h5>', html)
#       self.assertIn('Integration of Twitter API', html)

#       self.assertIn('<div id="slideshows"', html)
#       self.assertIn('Try NewsTracker with our dummy account', html)
#       self.assertIn('<div class="carousel slide" data-ride="carousel">', html)

def test_create_user(self):
    """Simulates POST Request to create a new user"""
    with app.test_client() as client:
        res = client.post("/register", follow_redirects=True, data=registration_data)
        html = res.get_data(as_text=True)
        self.assertEqual(res.status_code, 200)
        self.assertIn('Congratulations! You have successfully created an account.', html)

def test_login_user(self):
    """Simulates POST Request to login a user"""
    with app.test_client() as client:
        user_creds = User.query.get(self.user_id)
        res = client.post("/login", follow_redirects=True, data={'username': user_creds.username,
```

```
            'password': "test4444"})
            html = res.get_data(as_text=True)
            self.assertEqual(res.status_code, 200)
            self.assertIn('Credentials verified. You are now logged in.', html)

    def test_login_user_invalid(self):
        """Simulates an invalid POST Request to register a user"""
        with app.test_client() as client:
            res = client.post("/login", follow_redirects=True, data={'username':'testuser',
            'password':'pass1',
            'email':'test@email.com',
            'first_name':'test',
            'last_name':'user'})
            html = res.get_data(as_text=True)
            self.assertIn('Invalid username or password. Please try again', html)


class With_User_Without_Data(TestCase):
    def setUp(self):
        """Register sample user. (No Query or Story data)"""
        #clear previous
        User.query.delete()
        #register user
        user = User.register(
            "testuser", "test4444", "test@test.com", "test", "user")
        db.session.add(user)
        db.session.commit()
        self.user_id = user.id

    def tearDown(self):
        db.session.rollback()
```

```
# def test_home_page(self):
#     """Simulates non-user selecting headlines tab"""
#     with app.test_client() as client:
#         with client.session_transaction() as change_session:
#             change_session[CURR_USER_KEY] = self.user_id
#         res = client.get("/")
#         self.assertEqual(res.status_code, 200)
#         html = res.get_data(as_text=True)
#         #tests all categories are appearing
#         self.assertIn("Business", html)
#         self.assertIn("Science", html)
#         self.assertIn("Health", html)
#         self.assertIn("Entertainment", html)
#         self.assertIn("Sports", html)
#         self.assertIn("Technology", html)

#         #tests html elements and classes
#         self.assertIn('What can I do with NewsTracker?</h3>', html)
#         self.assertIn('Non-Users</h4>', html)
#         self.assertIn('Get Headlines</h5>', html)
#         self.assertIn('Search by Keyword</h5>', html)
#         self.assertIn('Browse by Category</h5>', html)
#         self.assertIn('Users</h4>', html)
#         self.assertIn('Detailed Search</h5>', html)
#         self.assertIn('Sentiment Analysis Tools</h5>', html)
#         self.assertIn('Saved Stories</h5>', html)
#         self.assertIn('Upcoming Features</h4>', html)
#         self.assertIn('Amplified User Features</h5>', html)
#         self.assertIn('Integration of Twitter API', html)
```

```
#       self.assertIn('<div id="slideshows"', html)
#       #Asserts that option for dummy account is NOT available
#       self.assertNotIn('Try NewsTracker with our dummy account"', html)
#       self.assertIn('<div class="carousel slide" data-ride="carousel">', html)

def test_nav_bar(self):
    """Asserts that user features are now present in navbar"""
    with app.test_client() as client:
        with client.session_transaction() as change_session:
            change_session[CURR_USER_KEY] = self.user_id
        res = client.get("/headlines")
        self.assertEqual(res.status_code, 200)
        html = res.get_data(as_text=True)
        self.assertIn('My Queries', html)
        self.assertIn('My Stories', html)
        self.assertIn('Detailed Search', html)
        self.assertIn('Logout', html)
        #asserts that non user navigation items are not present
        self.assertNotIn('Login', html)
        self.assertNotIn('Register', html)

def test_no_user_data(self):
    """Asserts that no stories message is present"""
    with app.test_client() as client:
        with client.session_transaction() as change_session:
            change_session[CURR_USER_KEY] = self.user_id
        res = client.get("/user/saved")
        self.assertEqual(res.status_code, 200)
        html = res.get_data(as_text=True)
        self.assertIn('<span class="no-stories">', html)
        """Asserts that no queries message are present"""
```

```
            self.assertIn('You currently have no saved queries', html)

    def test_logout(self):
        with app.test_client() as client:
            with client.session_transaction() as change_session:
                change_session[CURR_USER_KEY] = self.user_id
            res = client.get("/logout", follow_redirects=True)
            self.assertEqual(res.status_code, 200)
            html = res.get_data(as_text=True)
            self.assertIn('You have successfully logged out', html)



class With_User_With_Data(TestCase):
    def setUp(self):
        """Register sample user and attach data"""
        Query.query.delete()
        Story.query.delete()
        User.query.delete()

        user = User.register(
            "testuser", "test4444", "test@test.com", "test", "user")
        db.session.add(user)
        db.session.commit()

        story = Story(
        headline= 'Leaked documents show the hoops Roblox jumped through to do business in China',
        source= 'Engadget',
        author = 'Stevy Hawks',
        content = "In late June, Blizzard delayed",
        url= 'https://www.engadget.com/roblox-china-documents-172350532.html',
```

```
    image= 'https://s.yimg.com/os/creatr-uploaded-images/2022-07/804e2750-0c3c-11ed-97f7-998944b9b6f4',
    published_at= "2022-05-08",
    id= '75f3410055',
    user_id = user.id)

    db.session.add(story)
    db.session.commit()

    query = Query(
            name = "United Kingdom",
            user_id = user.id,
            source = 'bbc-news',
            keyword="UK",
            quantity=12,
            language='en',
            default=True,
            type="detailed_search",
            sa='polarity',
            sort_by='relevancy',
            date_from="2022-07-22",
            date_to="2022-09-08")
    #todo: change the date_from to automatically be one month out from present day, newstracker free tier only allows for one month in the past

    db.session.add(query)
    db.session.commit()

    self.user_id = user.id
    self.story_id = story.id

  def tearDown(self):
```

```
        db.session.rollback()


    def test_saved_stories(self):
        """Simulates user/saved route to ensure demo story in present"""
        with app.test_client() as client:
            with client.session_transaction() as change_session:
                change_session[CURR_USER_KEY] = self.user_id
            story = Story.query.get(self.story_id)
            res = client.get("/user/saved")
            html = res.get_data(as_text=True)
            self.assertEqual(res.status_code, 200)
            self.assertIn('Leaked documents', html) # portion of headline from sample story attached to
class and saved in db
            self.assertIn(story.headline, html)
            self.assertIn(story.content, html)


    def test_default_query_redirect(self):
        """Simulates headlines route to ensure default query works"""
        with app.test_client() as client:
            with client.session_transaction() as change_session:
                change_session[CURR_USER_KEY] = self.user_id
            res = client.get("/headlines")
            self.assertEqual(res.status_code, 302)


    def test_default_query_results(self):
        """Simulates headlines redirect successfull response with stories"""
        with app.test_client() as client:
            with client.session_transaction() as change_session:
                change_session[CURR_USER_KEY] = self.user_id
        res = client.get("/headlines", follow_redirects=True)
        html = res.get_data(as_text=True)
```

```
        self.assertEqual(res.status_code, 200)
        self.assertIn('UK', html) # replace this with flash message confirming default q has been selected


    def test_get_detailed_search(self):
        """Simulates search route"""
        with app.test_client() as client:
            with client.session_transaction() as change_session:
                change_session[CURR_USER_KEY] = self.user_id
            res = client.get("/search")
            html = res.get_data(as_text=True)
            self.assertEqual(res.status_code, 200)
            self.assertIn('<form method="POST" novalidate>', html)


    def test_query_name(self):
        with app.test_client() as client:
            with client.session_transaction() as change_session:
                change_session[CURR_USER_KEY] = self.user_id
            res = client.get("/search") #route is arbitrary since we are checking navbar elements
            html = res.get_data(as_text=True)
            self.assertIn("United Kingdom", html)


    def test_cant_save_dups(self):
        #FIGURE OUT HOW TO TEST ERRORS CORRECTLY
        with app.test_client() as client:
            with client.session_transaction() as change_session:
                change_session[CURR_USER_KEY] = self.user_id
                story = {
                'headline': 'Leaked documents show the hoops Roblox jumped through to do business in
China',
                'source': 'Engadget',
                "description": "something, something",
```

```
            'author':  'Stevy Hawks',
            'content':  "In late June, Blizzard delayed",
            'url': 'https://www.engadget.com/roblox-china-documents-172350532.html',
            'image':'https://s.yimg.com/os/creatr-uploaded-images/2022-07/804e2750-0c3c-11ed-97f7-998944b9b6f4',
            'published_at': "2022-05-08",
            'id': '75f3410055',
            'user_id': self.user_id}
            change_session['results'] = [story]


        test = Story.query.get(self.user_id)
        self.assertRaises(exc.SQLAlchemyError, client.post, f"/story/{self.story_id}/save_story")
        with self.assertRaises(exc.SQLAlchemyError):
            client.post(f"/story/{self.story_id}/save_story")


        #https://stackoverflow.com/questions/19289291/what-is-a-proper-way-to-test-sqlalchemy-code-that-throw-integrityerror
```