```
from flask import (Flask, g, render_template, flash, redirect, url_for, abort)
from flask_bcrypt import check_password_hash
from flask_login import (LoginManager, login_user, logout_user,
                         login_required, current_user)

import forms
import models

# Web Environment Variable Settings
HOST = '0.0.0.0'
PORT = 8000
DEBUG = True  # Set to FALSE, once web app is in production.

# Creates a Flask Object
app = Flask(__name__)
# For web session use.
app.secret_key = 'lkjasd.8103owerIDJ484*$K#*@(kjsdf)sdf93l)9pa-1094lkf'

# LoginManager() from Flask-Login provides user session management for Flask.
login_manager = LoginManager()
login_manager.init_app(app)
login_manager.login_view = 'login'


@login_manager.user_loader
def load_user(userid):
    try:
        # Checks if user id is the same as the one in the database.
        return models.User.get(models.User.id == userid)
    except models.DoesNotExist:
        # Exception comes from Peewee library.
        return None


@app.before_request
def before_request():
    """Connects to the database before each request."""
    g.db = models.APP_DATABASE
    g.db.connect()
    g.user = current_user


@app.after_request
def after_request(response):
    """Closes the database connection after each request."""
    # "response" is necessary object for render_templates, redirects, etc...
    g.db.close()
    return response


# Used in search form
def get_search_query(category, field):
    """Selects the proper search peewee search query based off what the User
        has selected for search."""
    if category == 'title':
        return models.BookPost.select().where(models.BookPost.title == field).
```

```
            limit(250)
    elif category == 'isbn':
        if field.isdigit():
            return models.BookPost.select().where(models.BookPost.isbn == field
                ).limit(250)
        else:
            return models.BookPost.select().where(models.BookPost.user == 0)
    elif category == 'course':
        return models.BookPost.select().where(models.BookPost.course == field).
            limit(250)
    elif category == 'major':
        return models.BookPost.select().where(models.BookPost.major == field).
            limit(250)
    elif category == 'user':
        try:
            user = models.User.get(models.User.username == field)
            return models.BookPost.select().where(models.BookPost.user == user.
                id).limit(250)
        except models.DoesNotExist:
            return models.BookPost.select().where(models.BookPost.user == 0)


# ------ #
# ROUTES #
# ------ #


@app.route('/', methods=('GET', 'POST'))
def index():
    form = forms.SearchForm()
    if form.validate_on_submit():
        search_results = get_search_query(form.category.data, form.search_field
            .data)
        return render_template('search.html', form=form, results=search_results
            )
    return render_template('index.html', form=form)


@app.route('/login', methods=('GET', 'POST'))
def login():
    form = forms.LoginForm()
    if form.validate_on_submit():
        try:
            # Peewee Query
            user = models.User.get(models.User.email == form.email.data)
        except models.DoesNotExist:
            flash("Your email or password do not match!", "error")
        else:
            if check_password_hash(user.password, form.password.data):
                # Creates a session, generates cookie in user's browser.
                login_user(user)
                flash("Welcome back!", "success")
                return redirect(url_for('account'))
            else:
                flash("Your email or password do not match!", "error")
    return render_template('login.html', form=form)
```

```
@app.route('/logout')
@login_required
def logout():
    # Deletes the session, also deletes the session cookie in user's browser.
    logout_user()
    flash("You've been logged out!", "success")
    return redirect(url_for('index'))


@app.route('/register', methods=('GET', 'POST'))
def register():
    form = forms.RegistrationForm()
    if form.validate_on_submit():
        models.User.create_user(
            username=form.username.data,
            email=form.email.data,
            password=form.password.data
        )
        flash("Thanks for registering. Please login with your shiny new
            account!", "success")
        logout_user()
        return redirect(url_for('login'))
    return render_template('register.html', form=form)


@app.route('/account', methods=('GET', 'POST'))
@login_required
def account():
    # "current_user" comes from flask login
    user = current_user
    bookposts = models.BookPost.select().where(models.BookPost.user == user.id)

    conversations = models.Message.select().distinct().order_by(-models.Message
        .timestamp)

    conversation_form = forms.ConversationForm()
    if conversation_form.validate_on_submit():
        convo_user = models.User.get(models.User.username == conversation_form.
            user.data)
        received_messages = models.Message.select().where(
            models.Message.from_user == convo_user.id,
            models.Message.to_user == user.id
        )
        sent_messages = models.Message.select().where(
            models.Message.to_user == convo_user.id,
            models.Message.from_user == user.id
        )
        return render_template('conversation.html',
                               convo_user=convo_user,
                               received_messages=received_messages,
                               sent_messages=sent_messages
                               )
    return render_template('account.html',
                           user=user,
```

```
                            conversations=conversations,
                            bookposts=bookposts,
                            conversation_form=conversation_form
                            )


@app.route('/update', methods=('GET', 'POST'))
@login_required
def update_account():
    update_username_form = forms.UpdateUsername()
    update_email_form = forms.UpdateEmail()
    update_password_form = forms.UpdatePassword()
    user = current_user
    if update_username_form.validate_on_submit():
        models.User.update(username=update_username_form.username.data).where
            (models.User == user.id).execute()
        flash("Your username has been updated.", "success")
        return redirect(url_for('account'))
    if update_email_form.validate_on_submit():
        models.User.update(email=update_email_form.email.data).where(models.
            User == user.id).execute()
        flash("Your email has been updated.", "success")
        return redirect(url_for('account'))
    if update_password_form.validate_on_submit():
        models.User.update(password=update_password_form.password.data).where
            (models.User == user.id).execute()
        flash("Your password has been updated.", "success")
        return redirect(url_for('account'))
    return render_template('update.html',
                            user=user,
                            update_username_form=update_username_form,
                            update_email_form=update_email_form,
                            update_password_form=update_password_form
                            )


@app.route('/new_message', methods=('GET', 'POST'))
@login_required
def message():
    form = forms.MessageForm()
    if form.validate_on_submit():
        models.Message.create(
            from_user=current_user.id,
            to_user=models.User.get(models.User.username == form.to_user.data),
            content=form.content.data.strip()
        )
        flash("Message has been sent!", "success")
        return redirect(url_for('account'))
    return render_template('message.html', form=form)


@app.route('/bookpost', methods=('GET', 'POST'))
@login_required
def book_post():
    form = forms.BookPostForm()
    if form.validate_on_submit():
```

```
            flash("Thanks for creating your Book Post!", "success")
            models.BookPost.create_book_post(
                user=g.user._get_current_object(),
                title=form.title.data,
                isbn=form.isbn.data,
                course=form.course.data,
                major=form.major.data,
                condition=form.condition.data,
                comment=form.comment.data
            )
            return redirect(url_for('account'))
        return render_template('bookpost.html', form=form)


    @app.route('/search', methods=('GET', 'POST'))
    def search():
        form = forms.SearchForm()
        if form.validate_on_submit():
            search_results = get_search_query(form.category.data, form.search_field
                .data)
            return render_template('search.html', form=form, results=search_results
                )
        return render_template('search.html', form=form)


    @app.route('/browse', methods=('GET', 'POST'))
    def browse():
        results = models.BookPost.select().limit(250).order_by('-creation_date')
        return render_template('browse.html', results=results)


    @app.route('/survey', methods=('GET', 'POST'))
    @login_required
    def survey():
        form = forms.SurveyForm()
        if form.validate_on_submit():
            # add the user and the post you're rating
            models.Survey.rate_user(
                user=models.User.get(models.User.username == form.user.data),
                rating=form.rating.data,
                comment=form.comment.data
            )
            flash("Rating received", "success")
        return render_template('survey.html', form=form)


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


    # Begins running Flask Web Application
    if __name__ == '__main__':
        models.initialize()
        app.run(debug=DEBUG, host=HOST, port=PORT)
```