```python
from flask_sqlalchemy import SQLAlchemy
from sqlalchemy.orm import relationship
from app import app, db
from werkzeug.security import generate_password_hash, check_password_hash
from datetime import datetime
from app.form_submissions import get_username, validate_phone_number
import csv


# INITS THE DBs for USERs
def add_postings(file):
    list = []
    key_list = ['userid', 'title', 'description', 'price', 'category', 'contactmethod']
    with open(file, 'r') as csv_file:
        reader = csv.DictReader(csv_file, delimiter=',', fieldnames=key_list)
        for row in reader:
            new_item = {}
            for key in key_list:
                new_item[key] = row[key]
            list.append(new_item)
    for value in list:
        tags = value['title'].split()
        tags = ','.join(tags)
        newPosting = Posting(
            userid          = value['userid'],
            date            = datetime.now(),
            title       = value['title'],
            description    = value['description'],
            price       = value['price'],
            category    = value['category'],
            contactmethod   = value['contactmethod'],
            tags            = tags
        )
        db.session.add(newPosting)
        db.session.commit()


def add_users(file):
    list = []
    key_list = ['phonenumber', 'email', 'personalemail', 'password', 'bio']
    with open(file, 'r') as csv_file:
        reader = csv.DictReader(csv_file, delimiter=',', fieldnames=key_list)
        for row in reader:
            new_item = {}
            for key in key_list:
                new_item[key] = row[key]
            new_item['phonenumber'] = validate_phone_number(new_item['phonenumber'])[1]
            new_item['username'] = get_username(new_item['email'])
            list.append(new_item)
    for value in list:
        newUser = User(
            username            = value['username'][1],
            email               = value['email'],
            personalemail       = value['personalemail'],
            password            = generate_password_hash(value['password']),
            phonenumber         = value['phonenumber'],
            bio                 = value['bio'],
            rating              = 5,
            numRatings          = 0
        )
        db.session.add(newUser)
        db.session.commit()


@app.cli.command('initdb')
def initdb_command():
    # wipeout
    db.drop_all()
    db.create_all()

    add_users("sampleUser.csv")
    # add some default data
    # db.session.add(User(username='jmd230', email="jmd230@pitt.edu", password=generate_password_hash('pass'), phonenumber='4121234567', personalemail='jordanmdeller@gmail.com', bio='Serious offers only', rating=2.51, numRatings=10))
    # db.session.add(User(username='admin', email="admin@pitt.edu", password=generate_password_hash('foobiz'), phonenumber='2341172381', personalemail='admin@admin.com', bio='I am an admin. This account is used to manage and test out the APP!', rating=5, numRatings=1))
    # db.session.add(User(username='tester1', email="tester1@pitt.edu", password=generate_password_hash('foobar'), phonenumber='2456734224', personalemail='tester1@gmail.com', bio='Tester is testing account for testing...', rating=3, numRatings=10))

    add_postings("postingsData.csv")

    db.session.add(Posting(userid=1, date=datetime.now(), title='Cool Book', description='Very good quality, barely used.', price=50.00, category='Textbooks', contactmethod='email', tags='book'))
    db.session.add(Posting(userid=2, date=datetime.now(), title='Brown couch', description='No signs of wear.', price=100.00, category='Furniture', contactmethod='phonenumber', tags='furniture, couch, seating, brown, comfy'))
    db.session.add(Posting(userid=2, date=datetime.now(), title='Cheap Book', description='Great quality.', price=20.00, category='Textbooks', contactmethod='personalemail', tags='book'))

    db.session.commit()

    print('Initialized the database.')


class User(db.Model):
    userid      = db.Column(db.Integer, primary_key = True)
    username      = db.Column(db.String(24), nullable = False)
    email       = db.Column(db.String(80), unique=True, nullable = False)
    # hashed password is ~100 chars ALWAYS
    password    = db.Column(db.String(128), nullable = False)
    phonenumber   = db.Column(db.String(64), nullable = False)
    personalemail   = db.Column(db.String(80), nullable = False)
    bio         = db.Column(db.String(250), nullable = False)
    rating      = db.Column(db.Float(2), nullable = False)
    numRatings    = db.Column(db.Integer, nullable = False)
    postings        = relationship("Posting", cascade="all,delete", backref="User")

    def __repr__(self):
        return '<User {}>'.format(self.username)


class Posting(db.Model):
    postid      = db.Column(db.Integer, primary_key = True)
    userid      = db.Column(db.Integer, db.ForeignKey("user.userid"))
    date      = db.Column(db.Date, nullable = False)
    title       = db.Column(db.String(30), nullable = False)
    description   = db.Column(db.String(250), nullable = False)
    price       = db.Column(db.Integer, nullable = False)
    category    = db.Column(db.String(80), nullable = False)
    contactmethod   = db.Column(db.String(80), nullable = True)
    tags      = db.Column(db.String(1000), nullable = True)
    claims        = relationship("Claim", cascade="all,delete", backref="Posting")

    def __repr__(self):
        return '<Posting {}: "{}">'.format(self.postid, self.title)
```

```

class Claim(db.Model):
    __table_args__ = (
        db.UniqueConstraint('postid', 'sellerid', 'buyerid', 'usersubmitted', name='unique_claim_buyer_seller'),
    )
    claimid     = db.Column(db.Integer, primary_key = True)
    postid      = db.Column(db.Integer, db.ForeignKey("posting.postid"))
    sellerid    = db.Column(db.Integer, db.ForeignKey("user.userid"))
    buyerid     = db.Column(db.Integer, db.ForeignKey("user.userid"))
    usersubmitted = db.Column(db.Integer, db.ForeignKey("user.userid"))
    date      = db.Column(db.Date, nullable = False)
    Rating          = db.Column(db.Integer, nullable = False)

    def __repr__(self):
        return '<Claim {}: "{}">'.format(self.claimid)


class Transaction(db.Model):
    transactionid = db.Column(db.Integer, primary_key = True)
    date            = db.Column(db.Date, nullable = False)
    claimidseller = db.Column(db.Integer, db.ForeignKey("claim.claimid"), nullable = False)
    claimidbuyer  = db.Column(db.Integer, db.ForeignKey("claim.claimid"), nullable = False)

    def __repr__(self):
        return '<Transaction {}: "{}">'.format(self.transactionid)


class ArchivedPosting(db.Model):
    __table_args__ = (
        db.UniqueConstraint('postid', 'buyerid', 'archivedpostid', 'sellerid', name='unique_archive_posting_constraint'),
    )
    archivedpostid  = db.Column(db.Integer, primary_key = True)
    transactionid   = db.Column(db.Integer, db.ForeignKey('transaction.transactionid'), nullable = True)
    postid      = db.Column(db.Integer, nullable = False)
    buyerid         = db.Column(db.Integer, db.ForeignKey("user.userid"), nullable = True)
    sellerid        = db.Column(db.Integer, db.ForeignKey("user.userid"), nullable = True)
    date      = db.Column(db.Date, nullable = False)
    title       = db.Column(db.String(80), nullable = False)
    description   = db.Column(db.String(250), nullable = True)
    price       = db.Column(db.Integer, nullable = False)
    category    = db.Column(db.String(80), nullable = False)
    contactmethod   = db.Column(db.String(80), nullable = True)
    tags      = db.Column(db.String(1000), nullable = True)

    def __repr__(self):
        return '<Posting {}: "{}">'.format(self.postid, self.title)

```