```
import datetime

from flask_bcrypt import generate_password_hash
from flask_login import UserMixin
from peewee import *

# Creates Database Object.
APP_DATABASE = SqliteDatabase('booki.sqlite3')


# "Model" comes from peewee
class User(UserMixin, Model):
    username = CharField(unique=True)
    email = CharField(unique=True)
    password = CharField(max_length=100)
    joined_at = DateTimeField(default=datetime.datetime.now)

    # Leaving admin feature as an option. If added later, then a new User Table
        must be created in the database.
    # is_admin = BooleanField(default=False)

    class Meta:
        database = APP_DATABASE
        order_by = ('-joined_at',)

    @classmethod
    def create_user(cls, username, email, password):
        try:
            with APP_DATABASE.transaction():
                cls.create(
                    username=username,
                    email=email,
                    password=generate_password_hash(password)
                    # Add "is_admin" to the method, and add a comma after
                        password if created.
                    # is_admin=is_admin
                )
        except IntegrityError:
            raise ValueError("Email and/or Username is already in use, please
                try again!", "success")

    def get_sent_messages(self):
        return Message.select().where(Message.from_user == self)

    def get_received_messages(self):
        return Message.select().where(Message.to_user == self)


class BookPost(Model):
    user = ForeignKeyField(
        rel_model=User,
        related_name='bookposts'
    )
    title = CharField()
    isbn = IntegerField(unique=True)
    course = CharField()
```

```
        major = CharField()
        condition = CharField()
        comment = CharField()
        creation_date = DateTimeField(default=datetime.datetime.now)
        is_available = BooleanField(default=True)

        class Meta:
            database = APP_DATABASE
            order_by = ('-creation_date',)

        @classmethod
        def create_book_post(cls, user, title, isbn, course, major, condition,
            comment):
            try:
                with APP_DATABASE.transaction():
                    cls.create(
                        user=user,
                        title=title,
                        isbn=isbn,
                        course=course,
                        major=major,
                        condition=condition,
                        comment=comment
                    )
            except IntegrityError:
                raise ValueError("This book already exists!")


    class Message(Model):
        from_user = ForeignKeyField(
            # Model that the foreign key points to.
            rel_model=User,
            # What the related model calls this model.
            related_name='relationships'
        )
        to_user = ForeignKeyField(
            # Model that the foreign key points to.
            rel_model=User,
            # What the related model calls this model.
            related_name='to_user'
        )
        content = TextField()
        timestamp = DateTimeField(default=datetime.datetime.now)

        class Meta:
            database = APP_DATABASE
            # Needs a comma because its a tuple.
            order_by = ('-timestamp',)


    class Survey(Model):
        user = ForeignKeyField(
            rel_model=User,
            related_name='user_being_rated'
        )
        rating = CharField()
```

```
        comment = CharField(max_length=250)
        time_of_rating = DateTimeField(default=datetime.datetime.now)

        class Meta:
            database = APP_DATABASE
            order_by = ('-time_of_rating',)

        @classmethod
        def rate_user(cls, user, rating, comment):
            try:
                with APP_DATABASE.transaction():
                    cls.create(
                        user=user,
                        rating=rating,
                        comment=comment
                    )
            except IntegrityError:
                raise ValueError("You already rated this user!")


    def initialize():
        APP_DATABASE.connect()
        APP_DATABASE.create_tables([User, BookPost, Message, Survey], safe=True)
        APP_DATABASE.close()
```