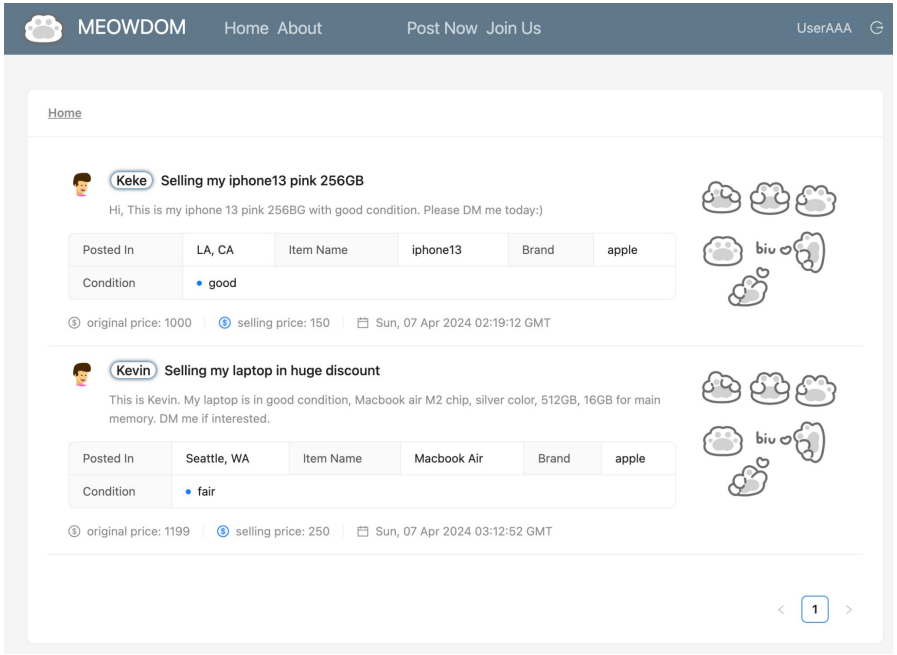
Meowdom

Liujia Yu, Jiawen Zhang, Kexin Sheng

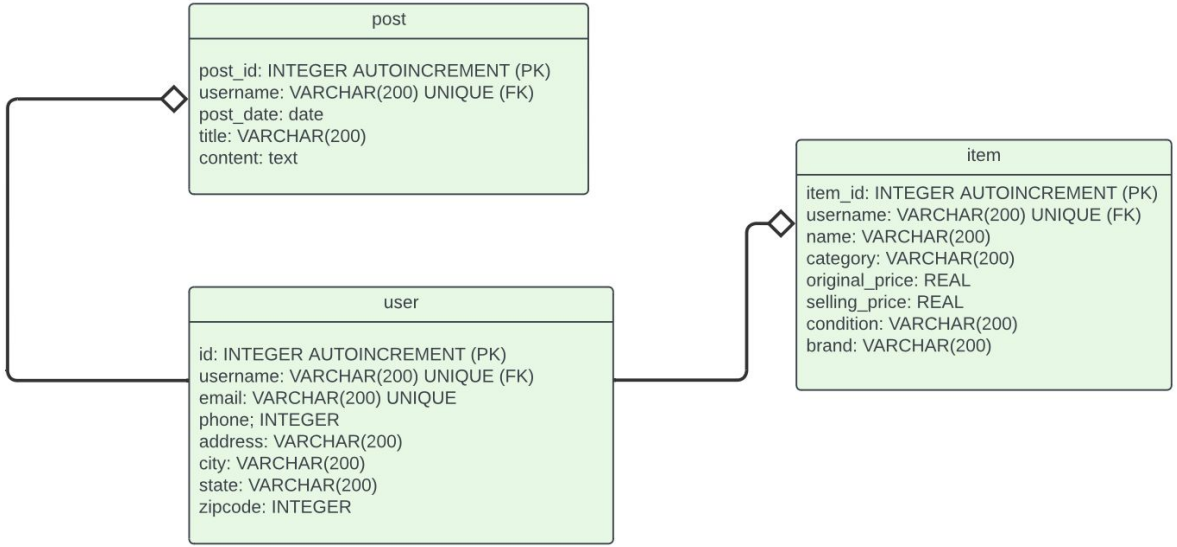
Introduction



- Mission: Second-Hand Transaction Platform
- User
  - JoinUs (Register)
  - Post
  - View
- Data Manager
  - Insert
  - Update
  - Delete
  - View

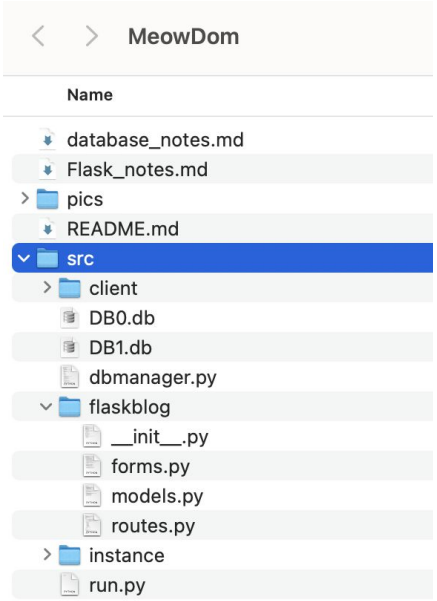
Database

- Database: 2 distributed database, DB0 & DB1
- Hash value: len(username)
- Database engine: SQlite



Backend

- Package: flask
- Files for backend:
  - _init_.py
  - models.py
  - forms.py
  - routes.py



Backend: _init_.py

```python
# initialize the application
from flask import Flask
from flask_sqlalchemy import SQLAlchemy

app = Flask(__name__)
app.config['SECRET_KEY'] = '5791628bb0b13ce0c676dfde280ba245'
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///database0.db'
app.config['SQLALCHEMY_BINDS'] = {
    'db0': 'sqlite:///database0.db',
    'db1': 'sqlite:///database1.db'
}

db = SQLAlchemy(app)
db0 = 'sqlite:///DB0.db'
db1 = 'sqlite:///DB1.db'
engine0 = db.create_engine(db0)
engine1 = db.create_engine(db1)

session0 = db.sessionmaker(bind=engine0)
session1 = db.sessionmaker(bind=engine1)


# function for return the database according to the hash value of the username
fionaz912
def find_db(username):
    if len(username) % 2 == 0:
        return session0()
    else:
        return session1()
```

- Initialize our APP
- Create two database engines connect to DB0 and DB1
- Define hash function to return the correct db

Backend: models.py

- Class:
  - Post
  - Item
  - User
- Function: create a post/item/user object and then insert to the post/item/user table in routes

```python
from datetime import datetime
from flaskblog import db


fionaz912 *
class User(db.Model):
    id = db.Column(db.Integer, primary_key=True, autoincrement=True)
    username = db.Column(db.String(200), unique=True, nullable=False)
    email = db.Column(db.String(200), unique=True, nullable=False)
    phone = db.Column(db.Integer, nullable=False)
    address = db.Column(db.String(200), nullable=False)
    city = db.Column(db.String(200), nullable=False)
    state = db.Column(db.String(200), nullable=False)
    zipcode = db.Column(db.Integer, nullable=False)
    posts = db.relationship('Post', backref='author', lazy=True)
    items = db.relationship('Item', backref='author', lazy=True)

    fionaz912
    def __repr__(self):
        return f"User('{self.username}', '{self.email}')"

fionaz912
class Post(db.Model):
    post_id = db.Column(db.Integer, primary_key=True, autoincrement=True)
    username = db.Column(db.String(200), db.ForeignKey('user.username'), unique=True, nullable=False)
    title = db.Column(db.String(200), nullable=False)
    post_date = db.Column(db.DateTime, nullable=False, default=datetime.utcnow)
    content = db.Column(db.Text, nullable=False)

    fionaz912
    def __repr__(self):
        return f"Post('{self.title}', '{self.date_posted}')"
```

Backend: forms.py

- Classes: RegistrationForm and PostForm (Flaskform)
- RegistrationForm:
  - Transform the data from website into correct data type
  - Validate unique email, username
- PostForm:
  - Transform the data from website into correct data type
  - Validate user has already registered

```python
class RegistrationForm(FlaskForm):
    username = StringField('Username',
                           validators=[DataRequired(), Length(min=2, max=200)])
    email = StringField('Email',
                        validators=[DataRequired(), Email()])
    phone = IntegerField('Phone', validators=[DataRequired()])
    address = TextAreaField('Address',  validators=[DataRequired()])
    city = StringField('City', validators=[DataRequired()])
    state = StringField('State', validators=[DataRequired()])
    zipcode = IntegerField('Zipcode', validators=[DataRequired()])
    submit = SubmitField('Sign Up')

    fionaz912
    def validate_username(self, username):
        user1 = session0().query(User).filter_by(username=username.data).first()
        user2 = session1().query(User).filter_by(username=username.data).first()

        # user = User.query.filter_by(username=username.data).first()
        if user1 or user2:
            raise ValidationError('That username is taken. Please choose a different one.')

    fionaz912
    def validate_email(self, email):
        user1 = session0().query(User).filter_by(email=email.data).first()
        user2 = session1().query(User).filter_by(email=email.data).first()
        # user = User.query.filter_by(email=email.data).first()
        if user1 or user2:
            raise ValidationError('That email is taken. Please choose a different one.')
```

Backend: routes.py

- Import classes from models, forms, import find_db in _init_.py
- Home: Display all posts in DB
- About: Introduction
- joinUs(register): Register as a member
    - Get data from web => Create a registrationForm object to correct data type => validate => Create a user object => Insert to DB
- postNow: Post items willing to sell
    - Get data from web => Create a postForm => validate => Create post&item objects => Insert to DB

```python
@app.route("/register", methods=['POST'])
def register():
    # Get registration data from request
    data = request.get_json()
    form = RegistrationForm(data=data)
    if form.validate_username(form.username) and form.validate_email(form.email):
        user = User(username=form.username.data, email=form.email.data, phone=form.phone.data,
                    address=form.address.data, city=form.city.data, state=form.state.data, zipcode=form.zipcode.data)
        cur_session = find_db(form.username.data)
        cur_session.add(user)
        cur_session.commit()
        return jsonify({"message": "Registration successful"}), 200
    else:
        return jsonify({"errors": form.errors}), 400
```

Database Manager

Data manager can **view, insert, update, delete** from the database

Command and options are encoded using numbers (1,2,3..), which is easier for manager to execute.

Example : data manager updates the item condition information

```
Welcome to Meowdom database management system!
Which command do you want to execute? 1. View 2. Insert, 3. Delete, 4. Update, 5.Quit
Please enter the number of the command that you want to execute: 4
Please enter the username that you want to update for: Kevin
Please select which table you want to update? 1. user 2. post, 3. item,4.Quit3
Please select the content you want to update on item table: 1.name 2.category 3.original price 4.selling price 5.condition 6.brand5
Enter the updated value:good
Updated successfully
```

Process: Choose command → Identify the user → Input updated value

Database Manager : Code display

- Overall logic → Four commands
  - Action 1:View
  - Action 2:Insert
  - Action 3:Delete
  - Action 4:Update
  - Action 5:Quit
- Identification function
  - check_username(username)
  - check_email(email)
- find_db(username)
- Connect (session)

```python
def check_username(username):
    cur_session = find_db(username)
    cur_session.execute('select * from user where username = ?', (str(username),))
    cur_user = cur_session.fetchone()
    if cur_user:
        return True
    return False

def check_email(email):
    session_0.execute('select * from user where email = ?', (str(email),))
    email_0 = session_0.fetchone()
    session_1.execute('select * from user where email = ?', (str(email),))
    email_1 = session_1.fetchone()
    if email_0 or email_1:
        return True
    return False

def find_db(username):
    if len(username) % 2 == 0:
        return session_0
    else:
        return session_1

def connect(session):
    if session == session_0:
        return connect_0
    else:
        return connect_1
```

```python
print("Welcome to Meowdom database management system! ")
while True:
    print("Which command do you want to execute? 1. View 2. Insert, 3. Delete, 4. Update, 5.Quit")
    action = int(input("Please enter the number of the command that you want to execute: "))
    if int(action) not in [1, 2, 3, 4]:
        print("Please enter a valid number: ")
    elif action == 1:
        view()
    elif action == 2:
        insert()
    elif action == 3:
        delete()
    elif action == 4:
        update()
    else:  # action == 4
        break
```

Database Manager : View

- View(): View all the lines from database 1 and database 0

```
def view():
    session_0.execute("select * from user")
    rows = session_0.fetchall()
    print('All users in DATABASE0:')
    for row in rows:
        print(row)

    session_0.execute("select * from post")
    rows = session_0.fetchall()
    print('All posts in DATABASE1:')
    for row in rows:
        print(row)

    session_0.execute("select * from item")
    rows = session_0.fetchall()
    print('All items in DATABASE0:')
    for row in rows:
        print(row)
```

```
    session_1.execute("select * from user")
    rows = session_1.fetchall()
    print('All users in DATABASE1:')
    for row in rows:
        print(row)

    session_1.execute("select * from post")
    rows = session_1.fetchall()
    print('All posts in DATABASE0:')
    for row in rows:
        print(row)

    session_1.execute("select * from item")
    rows = session_1.fetchall()
    print('All items in DATABASE1:')
    for row in rows:
        print(row)
```

Database Manager : Insert

- Insert () Insert a new data point to database
  - Action 1: insert into user table
  - Action 2:insert into post table
  - Action 3: insert into item table
- After Insert (), there are three options according to change the three tables
  - insert _user()
  - insert_item(username)
  - insert_post(username)

```
def insert():
    print("Which table do you want to insert into? 1.user, 2. post, 3.item")
    insert_table = int(input("Please enter a number: "))
    if insert_table not in [1, 2, 3, 4]:
        print("Please enter a valid number: ")
    elif insert_table == 1:
        print("Please enter the username: ")
```

```
def insert_item(username):
    name = str(input("Please enter the item name: "))
    category = str(input("Please enter the category: "))
    original_price = float(input("Please enter the original price: "))
    selling_price = float(input("Please enter the selling price: "))
    condition = str(input("Please enter the item condition: "))
    brand = str(input("Please enter the brand of the item: "))
    cur_session = find_db(username)
    try:
        cur_session.execute(
            "Insert into item (username, name, category, original_price, selling_price, condition,
            "values (?, ?, ?, ?, ?, ?, ?)",
            (username, name, category, original_price, selling_price, condition, brand))
        connect(cur_session).commit()
        print(f"Inserted the item {name} for the user {username} successfully")
```

Database Manager : Update

- update () update new value to database
  - Action 1: update on user table
  - Action 2: update on post table
  - Action 3: update on item table
- After update (), there are three options according to change the three tables
  - update_user(username)
  - update_item(username)
  - update_post(username)

```python
def update():
    while True:
        cur_user = str(input("Please enter the username that you want to update for: "))
        cur_session = find_db(cur_user)
        while check_username(cur_user):
            action = int(input("Please select which table you want to update? 1. user 2. post, 3. item,4.Quit"))
            username = cur_user

            if action == 1:
                update_user(username)
            elif action == 2:
                update_post(username)
            elif action == 3:
                update_item(username)
            else:
                #print('No table found, please go back to check table name again.')
                break
```

```python
def update_user(username):
    session = find_db(username)
    connection = connect(session)
    #choose update content
    action = int(input('Please select the content you want to update on user table: 1.phone 2.address 3.city 4.state 5.zipcode'))

    #enter updated value
    if action in (1, 5):
        updated_value =int(input('Enter the updated value:'))
    else:
        updated_value =str(input('Enter the updated value:'))

    action_dic = {1:"phone",2:"address",3:"city",4:"state",5: "zipcode"}
    try:
        session.execute(f'update user set {action_dic[action]} = ? where username = ?', (updated_value,username,))
        if session.rowcount > 0:
            connection.commit()
            print('Updated successfully')
        else:
            print('No content were updated.')

    except sqlite3.Error as e:
        print(f"An error occurred: {e}")
```

Database Manager : Delete

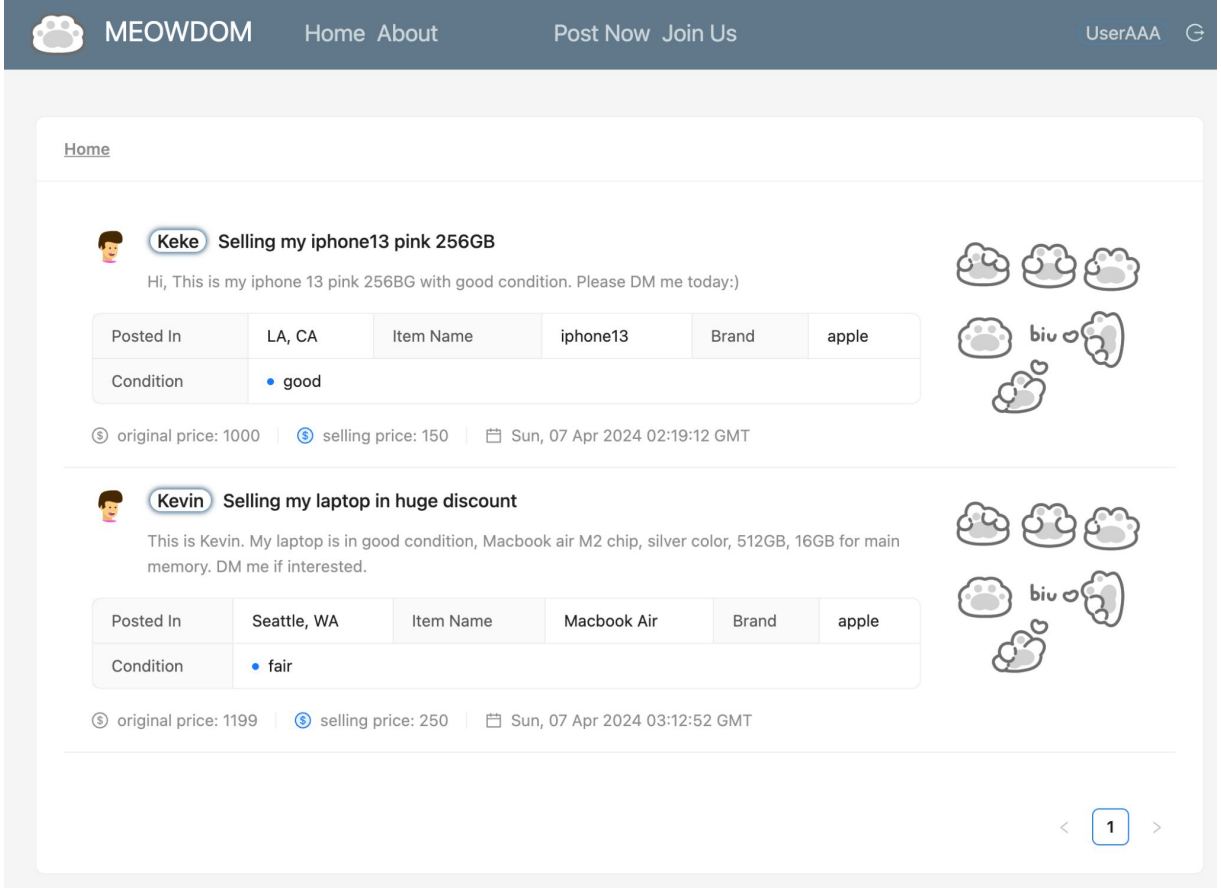
- delete () delete value by username
  - Action 1: delete all info
  - Action 2: delete a post for that user
  - Action 3: delete an item for that user
  - Action 4: quit
- After delete (), there are three options according to change the three tables
  - delete_user(cur_user)
  - delete_item(cur_user)
  - delete_post(cur_user)

```
def delete():
    cur_user = str(input("Please enter the username that you want to delete for: "))
    while not check_username(cur_user):
        cur_user = str(input("We cannot find this user in the database. Please find another one: "))
    print("Which cammand do you wan to execute?")
    print("1. Remove the user and all of their infomation from the database.")
    print("2. Delete a post for this user.")
    print("3. Delete an item for this user.")
    print("4. Quit")
    while True:
        cur_action = int(input("Please enter the number of execution: "))
        while cur_action not in [1, 2, 3, 4]:
            cur_action = int(input("Please enter a valid number of execution: "))
        if cur_action == 1:
            delete_user(cur_user)
        elif cur_action == 2:
            delete_post(cur_user)
        elif cur_action == 3:
            delete_item(cur_user)
        else:
            break
```

```
def delete_item(cur_user):
    print(f"Please review the items posted by the user {cur_user} and find the id of the item that you want to delete.")
    cur_session = find_db(cur_user)
    cur_session.execute("Select * from item where username = ?", (cur_user, ))
    items = cur_session.fetchall()
    header = []
    for i in cur_session.description:
        header.append(i[0])
    print(header)
    for i in items:
        print(i)
    cur_id = int(input(f"Please enter the item id that you want to delete for {cur_user}: "))
    valid_id = cur_session.execute("Select item_id from item where username=?", (cur_user, )).fetchall()
    valid_id_list = [i[0] for i in valid_id]
    while cur_id not in valid_id_list:
        cur_id = int(input(f'Please enter a valid item id: '))
    try:
        cur_session.execute('Delete From item where item_id = ?', (cur_id,))
        connect(cur_session).commit()
        print(f'Deleted the item with item_id {cur_id} for the user {cur_user} successfully!')
    except:
        print(f"Failed to remove the item with item_id {cur_id} for the user {cur_user}. Please try again later!")
```

Frontend

- React
- Styles:
  - Bootstrap
- UI & layout:
  - ant-design



Communicate with backend

- Proxy: port 5000
- fetch

PostNow/index.js | Register/index.js | package.json

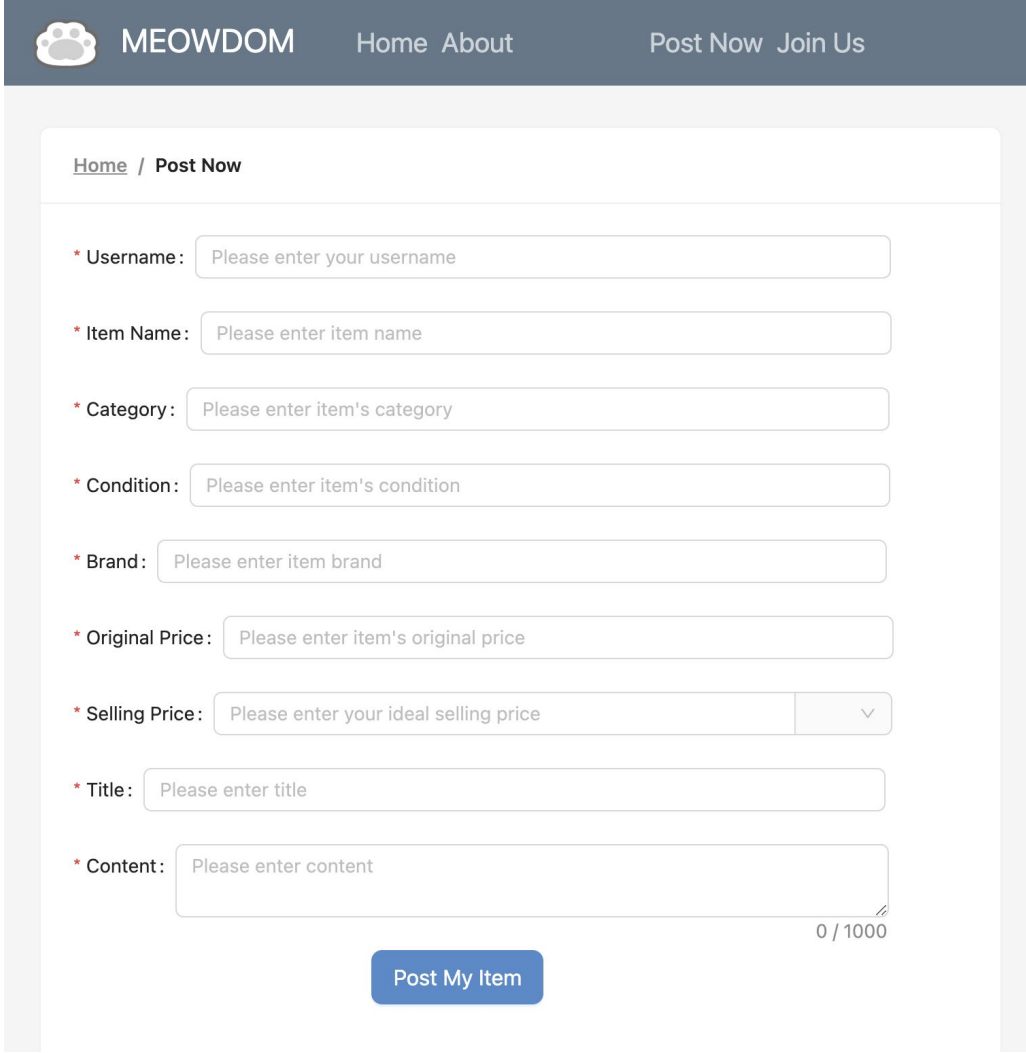
```
{
  "name": "client",
  "version": "0.1.0",
  "private": true,
  "proxy": "http://localhost:5000",
  "dependencies": {
    "@testing-library/jest-dom": "^5.17.0",
    "@testing-library/react": "^13.4.0",
    "@testing-library/user-event": "^13.5.0",
    "antd": "^5.16.2",
```

```
useEffect( effect: () : void => {
    try {
        async function fetchRegister() : Promise<void> {  Show usages  Liujia Yu
            fetch( input: "/register") Promise<Response>
                .then(
                    response : Response => response.json()
                ) Promise<any>
                .then(
                    data => {
                        setPosts(data);
                        // console.log("data.posts: \n", data.posts);
                    }
                )
        }

        fetchRegister()
    } catch (error) {
        console.log(error)
    }
}, deps: [])
```

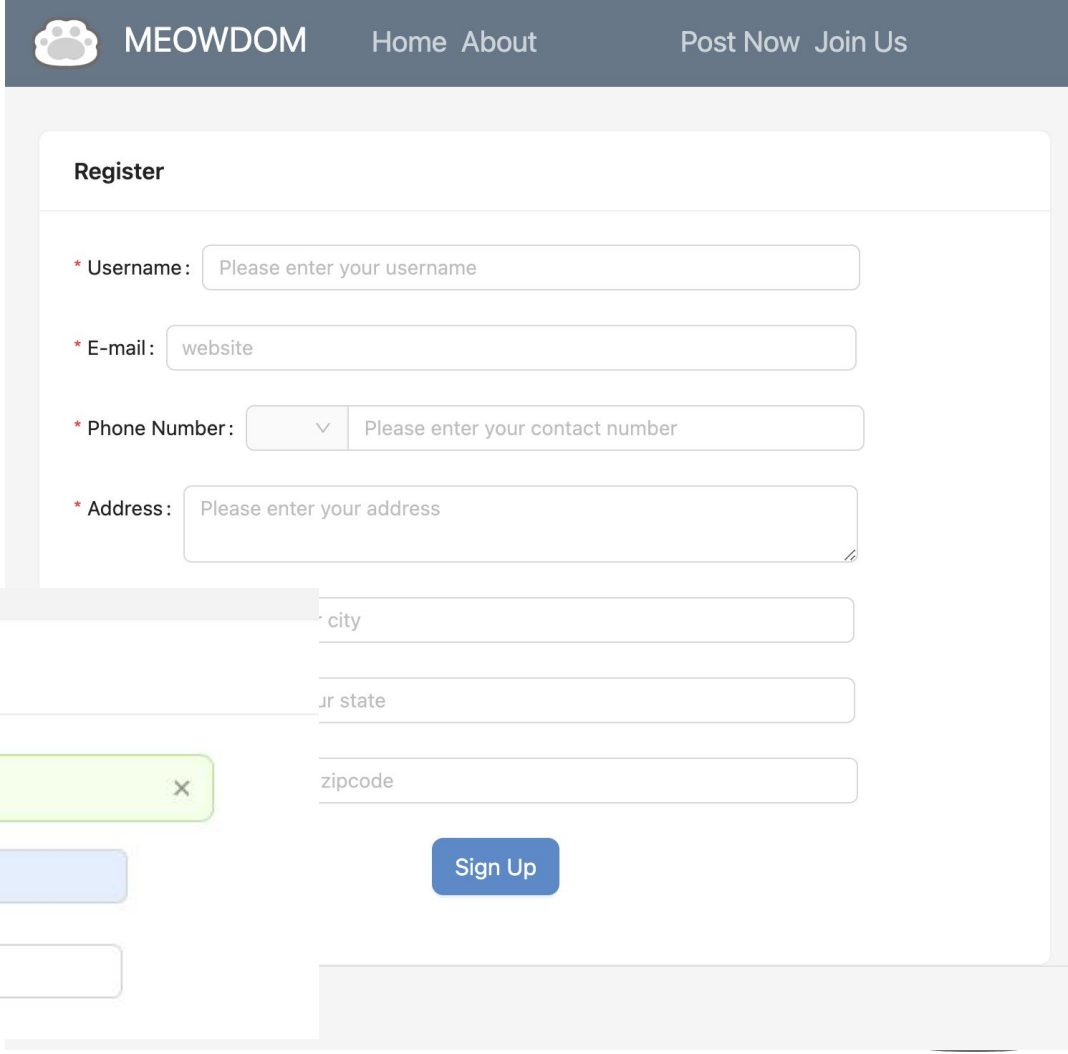
Frontend - post now

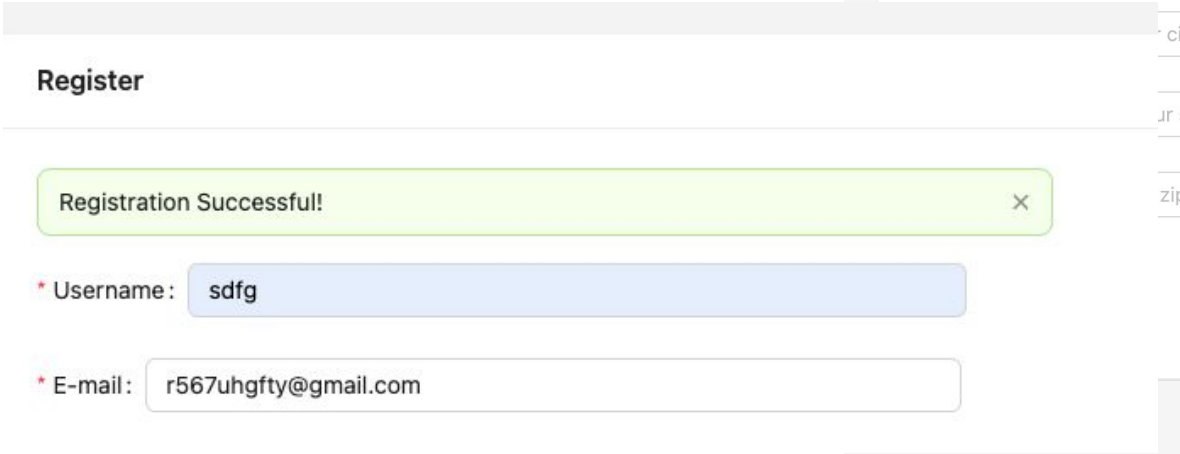
- React
- Styles:
  - Bootstrap
- UI & layout:
  - ant-design



Frontend - register

- React
- Styles:
  - Bootstrap
- UI & layout:
  - ant-design





Live Demo

User manual:

1. Backend: Under ‘Meowdom/src’ directory
    a. `python run.py`
2. Frontend: Under ‘Meowdom/src/client/src’ directory
    a. `npm install`
    b. `npm start`

Q&A

Thanks for watching!

Our code: https://github.com/LesleyYu/MeowDom/tree/flask

Appendix: Reference

1. Flask Tutorial (for backend): https://www.youtube.com/watch?v=MwZwr5Tvyxo&list=PL-osiE80TeTs4UjLw5MM6OjgkjFeUxCYH