

REST API : set of rules and conventions for building and interacting with web services.

- 1) Stateless: Allows RESTful web services to work seamlessly with HTTP protocol, Servers treat each request independently, All data needed to complete a requested action is sent with each request.
- 2) Can access it by visiting url or using curl/HTTPie from CLI, PATCH is used for updating part of a datum, API resources always have unique identifiers.

Network Security

- 1) Certificates issued mitigates(lower) the risk of adversary(other) intercepting public keys.
- 2) Confidentiality stands for adversary should not understand the message / Integrity stands for not modifying the message.
- 3) Salt : putting salt can prevent rainbow table attack.
- 4) HTTPS : protect against Man-in-the-middle attack (Protected with PKI), Eavesdropping // but not Cross-site scripting

JavaScript

JavaScript is a lightweight interpreted programming language, enables developers to create dynamic pages, runs natively in modern browsers, provides built-in functionality to alter the DOM of a web page.

```
@app.route('/pokedex', methods=['GET', 'POST'])
```

```
def pokedex():
    con = get_db_connection()
    # If the method is GET
    if request.method == 'GET':
        type = flask.request.form['type']
        health = flask.request.form['health']
        cur = con.execute(
            SELECT pokemon.name, type, health, attack.name
            FROM pokemon, pokedex_entries
            WHERE pokemon.type = ?
            AND pokemon.health > ?
            AND pokedex_entries.trainer_id = ?
            AND pokemon.name = pokedex_entries.poke_name,
            (type, health, session['trainer_id'],)
        )
        pokemon = cur.fetchall()
        trainer = get_trainer()
        context = {'trainer': trainer, 'pokemon_container': pokemon}
        return render_template('pokedex.html', **context)
```

```
# If the method is POST
elif request.method == 'POST':
```

```
poke_name = flask.request.form['name']
level = flask.request.form['level']

con.execute(
    "INSERT INTO pokedex_entries (name, level, trainer_id)
    VALUES (?, ?, ?) ",
    (poke_name, level, session['trainer_id'])
)

pokemon = get_pokemon(poke_name)
type = pokemon['type']
health = ""

return redirect(url_for('pokedex', type=type, health=health))
```

Trainer Name: Maritan

Pokemon Name:

Pokemon Level:

submit

Pokedex:

Name: Charmander

Type: fire

Attack name: Flamethrower

Health: 30

Name: Charizard

Type: fire

Attack name: Fire Blast

Health: 30

<body>

```
<p>Trainer Name: {{trainer.trainer_name}}</p>
<form action="{{url_for('pokedex')}}" method="post"
  enctype="multipart/form-data">
  <p>Pokemon Name:</p>
  <input type="text" name="name" required />
  <p>Pokemon Level:</p>
  <input type="text" name="level" required />
  <input type="submit" value="submit" />
</form>
```

```
<h1> Pokedex:</h1>
{% for pokemon in pokemon_container %}
<div>
  <p> Name: {{pokemon.name}} </p>
  <p> Type: {{pokemon.type}} </p>
  <p> Attack name: {{pokemon.attack_name}} </p>
  <p> health: {{pokemon.health}} </p>
</div>
{% endfor %}
</body>
```

Task Component

import React, { useState, useEffect } from 'react';

```
function Task({ taskId, description, dueDate, handleSubmission }) {
  return (
```

- 1) Need to render the button, the task with description & due dates
- 2) Need to handle the click (use handleSubmission)

```
<input type="submit" value="Done"
  onClick={e => handleSubmission(e, taskId)}
  id={taskId} />
  {description} Due date: {dueDate}
</>
);
}
```

```
const handleSubmission = (event, taskId) => {
```

SubmitTask(taskId);

TaskList Component

```
function TaskList() {
  const [tasks, setTasks] = useState([]);
  // state variable = tasks
  // takes in function used to update the state
  // initially set as an empty array
```

```
const submitTask = (taskId) => {
  1) Make REST API call to the server. -> To submit a POST request to submit the task
  2) Modify the corresponding state -> Delete the submitted task from the Tasks state
  fetch('/api/tasks/' + taskId, {
    method: "POST",
    // API for submitting a task
    // POST /api/task/<id>/
  })
  .then(() => {
    // This will filter out the item id == taskId
    const newTasks = tasks.filter(task => task.id !== taskId);
    setTasks(newTasks); // task 삭제
  })
  .catch(error) => {
    console.error("Error:", error);
  });
};
```

```
useEffect(() => {
```

```
Need to store all the tasks to tasks(state)
fetch('/api/tasks/') // fetch all the tasks from DB
.then(response => response.json())
.then(tasks => {
  setTasks(tasks);
})
.catch(err) => {
  console.log(err);
});
```

```
}, []);
```

TODO485

3 tasks left

Done 485 Project 3 Due Date: 2021-01-01

Done Email teammates Due Date: 2021-01-02

Done Study for midterm Due Date: 2021-01-03

```
return (
```

We need to render all the tasks using the Task component.
We have a state 'tasks': a list of tasks
Need to use map to render tasks.map(task) => (<Task />)

```
<div>
  <h1> TODO485 </h1>
  <div> {tasks.length} tasks left </div>
  {tasks.map(task) => (
    <Task
      key={task.id}
      taskId={task.id}
      description={task.description}
      handleSubmission={handleSubmission}
    />
  )}
</div>
```

```
);
```

```
}
```

export default List;

```
@bank485.app.route('/')
def show_index():
    connection = bank485.model.get_db()

    # Get relevant usernames for homepage
    users = []
    logname = flask.session["logname"]
    cur = connection.execute(
        "SELECT friends.friend, users.profile_picture "
        FROM (
            SELECT usernames as friend
            FROM friends
            WHERE username1 == ?
            UNION
            SELECT username1 as friend
            FROM friends
            WHERE username2 == ?
        ) friends
        JOIN users
        ON users.username == friends.friend ",
        (logname, logname, )
    )
    # cur = logname's friends
    for friend in cur.fetchall():
        users.append({'username': friend['friend'],
                     'profile_picture': friend['profile_picture']})

    cur = connection.execute(
        "SELECT users.profile_picture "
        FROM users
        WHERE username == ? ,
        (logname, )
    )
    # cur = myself
    users.append({'username': logname,
                 'profile_picture': cur.fetchone()[0]})
```

```
@market485.app.route('/api/goods/')
def get_goods():
    # Connect to database
    connection = market485.model.get_db()
```

```
cur = connection.execute(
    "SELECT *"
    FROM goods "WHERE available == 1"
)
goods = []
cur_good = {'title': good['title'],
            'id': good['id'],
            'description': good['des']}
resp = cur.fetchall()
for good in resp:
    goods.append(cur_good)
return flask.jsonify(goods)
```

```
@market485.app.route('/api/add_good/', methods=['POST'])
def add_good():
    connection = market485.model.get_db()
    title = flask.request.form["title"]
    description = flask.request.form["description"]
    seller_username = flask.request.form["seller_username"]
    price = flask.request.form["price"]
```

```
connection.execute(
    "INSERT INTO goods (title, description, seller, price, available)"
    "VALUES (?, ?, ?, ?, ?),"
    (title, description, seller_username, price, 1)
)
# 임의적인 Insert.

cur = connection.execute("SELECT MAX(id) as id FROM goods")
id_dict = cur.fetchone() or flask.abort(404)
cur = connection.execute(
    "SELECT * FROM goods WHERE id = ?",
    (id, )
)
# 1 good (max id)를 goods에 추가
query_result = cur.fetchone()
if query_result is None:
    flask.abort(404)
# url 설정
query_result['resource_url'] = f"/api/good/{id}/"
return flask.jsonify(query_result)
```

```
@market485.app.route('/api/buy_good/', methods=['POST'])
def buy_good():
    connection = market485.model.get_db()
```

good_id = flask.request.json.get("good_id")

connection.execute("SELECT * FROM goods WHERE id = ?", (good_id,))

"DELETE FROM goods WHERE id = ?"

```
# users contains current usernames and profile pictures for logged in user and their friends

# Get relevant transactions, return rendered "homepage" template
transactions = []
cur = connection.execute(
    "SELECT *"
    FROM transactions "
    WHERE send_username == ? OR receive_username == ? ",
    (logname, logname, )
)
# cur = user send it receive it transactions
resp = cur.fetchall()
for transaction in resp:
    cur_transaction = {'send_username': transaction['send_username'],
                      'receive_username': transaction['receive_username'],
                      'transaction_amount': transaction['transaction_amount']}
    transactions.append(cur_transaction)

for cur_friend in users:
    cur = connection.execute(
        "SELECT send_username, receive_username, amount, id "
        FROM transactions "
        WHERE is_public == 1
        AND (send_username == ? OR receive_username == ?) ",
        (cur_friend['username'], " ", )
    )
    # cur = transactions from public old user send it receive
    resp = cur.fetchall()
    for transaction in resp:
        if transaction not in transactions:
            transactions.append(cur_transaction)

context = {'users': users, 'transactions': transactions}
return flask.render_template("homepage.html", **context)
```

Add Transaction

Send or Request Money? ☐ send ☐ request

Other User:

Transaction amount:

Public Transaction? ☐ yes ☐ no

```
<body>
<h1>Users</h1>
{% for user in users %}
<div>
{{user.username}}

</div>
{% endfor %}
<a href="/transaction_form/">Add Transaction</a>
{% for transaction in transactions %}
<div> {{transaction.send_username}} sent {{(transaction.transaction_amount)}}
to {{(transaction.receive_username)}} </div>
{% endfor %}
</body>

<h2>Add Transaction</h2>
<form action="/transaction/" method="post" enctype="multipart/form-data">
<b>Send or Request Money?</b>
<input type="radio" name="send_or_request" value="send">send
<input type="radio" name="send_or_request" value="request">request

<b>Other User:</b>
<input type="text" name="other_user" value="">

<b>Transaction amount:</b>
<input type="text" name="amount" value="">

<b>Public Transaction?</b>
<input type="radio" name="is_public" value=1>yes
<input type="radio" name="is_public" value=0>no

<input type="submit" name="submit" value="submit"/>
</form>
```

```
const BUY_GOOD_URL = 'http://localhost:8000/api/buy_good/'
const ADD_GOOD_URL = 'http://localhost:8000/api/add_good/'
const FETCH_GOODS_URL = 'http://localhost:8000/api/goods/'
```

```
const Good = (props) => {
    let {description, price, title, onClick} = props

    return (
        <div>
            {title}: ${price} - {description}
            <button onClick={onClick}>buy</button>
        </div>
    )
}
```

```
const App = () => {
    var [goods, setGoods] = useState([]);
    var [showForm, setShowForm] = useState(false);
```

```
let fetchGoods = function() {
    fetch(FETCH_GOODS_URL)
    .then(response => response.json())
    .then(data => setGoods(data));
}
```

```
let buyGood = function(id) {
    let options = {
        method: 'POST',
        body: JSON.stringify({ "good_id": id }),
        headers: { 'content-type': 'application/json' }
    }
    fetch(BUY_GOOD_URL, options)
    .then(fetchGoods());
}
```

```
let handleAddGood = function(event) {
    event.preventDefault()
    let options = {
        method: 'POST'
        body: JSON.stringify({
            "title": event.target.title.value,
            "price": event.target.price.value,
            "description": event.target.description.value,
        })
        headers: { 'content-type': 'application/json' }
    }
    fetch(ADD_GOOD_URL, options)
    .then(fetchGoods());
    event.target.reset()
}
```

// Execute initial effects and return components, feel free to add any additional functions you may need in this box

```
useEffect(() => {
    fetchGoods()
}, []);

return (
    <div>
        <h1>Market 485</h1>
        {goods.map((good, i) => {
            return <Good key={i} {...good} onClick={() => buyGood(good.id)} />
        })}

        <button onClick={() => setShowForm(!showForm)}>
            {showForm ? 'Hide' : 'Show'}
        </button>

        {showForm &&
            <form onSubmit={handleAddGood}>
                <label>Title: <input type="text" name="title"/> </label>
            </form>
        }
    </div>
)
```

```
@bank485.app.route('/transaction/', methods=['POST'])
def post_transaction():
    connection = bank485.model.get_db()
```

```
# extract relevant form information
logname = flask.session["logname"]
other_user = flask.request.form["other_user"]
amount = float(flask.request.form["amount"])
send_or_request = flask.request.form

send_user = ""
receive_user = ""
if (send_or_request == 'send'):
    send_user = logname
    receive_user = other_user
if (send_or_request == 'request'):
    # check if transaction requirements met, add to DB if so
    cur = connection.execute(
        "SELECT *"
        FROM friends
        WHERE username1 == ? AND username2 == ?
        UNION
        SELECT * FROM friends
        (send_user, receive_user, receive_user, send_user, )
    )
    possible_friends = cur.fetchall()
    if (not possible_friends):
        return flask.redirect(flask.url_for("show_index"))
    cur = connection.execute(
        "SELECT account_balance FROM users
        WHERE username == ? , (send_user)"
    )
    cur_balance = cur.fetchone()[0]
    if (cur_balance - amount < 0):
        return flask.redirect(flask.url_for("show_index"))
    connection.execute(
        "INSERT INTO transactions (send, receive, amount, is_public)"
        "VALUES (?, ?, ?, ?),"
        (send, receive, amount, is_public)
    )
    return flask.redirect(flask.url_for("show_index"))
```