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.

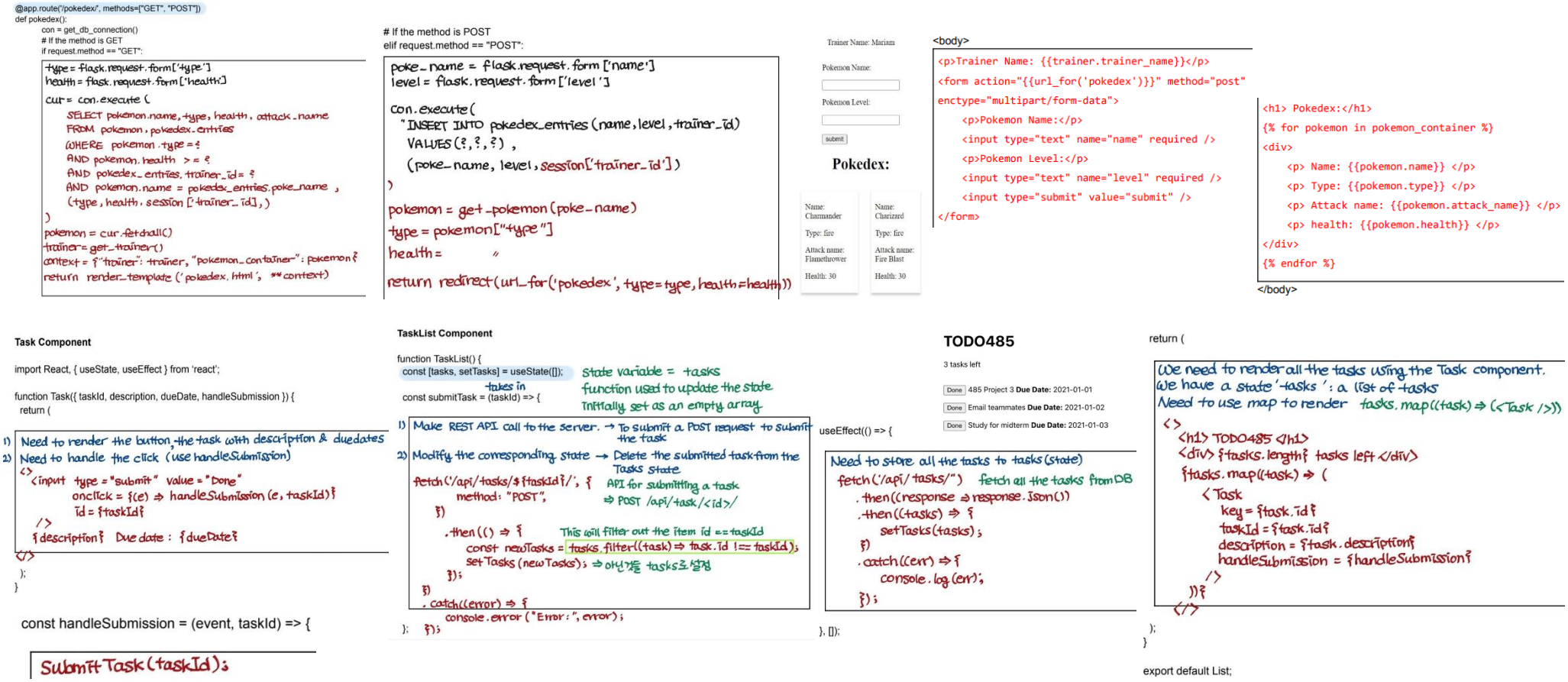
2) Confidentiality stands for adversary should not understand the message / Integrity stands for not modifying the message.

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

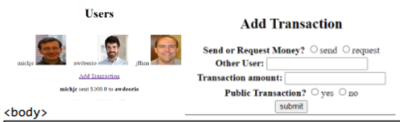
JavaScript is a lightweight interpreted programming language, enables developers to create dynamic pages, runs natively in modern browsers, provides built-in functionality to alter the DOM of a web page.



```python
@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 username1 as friend
         FROM friends
         WHERE username1 == ?
         UNION
         SELECT username1 as friend
        ) friends
       JOIN users
      " ON users.username == friends.friend ",
       (logname, logname, )
    )                                   # cur = logname's friends → users []
    for friend in cur.fetchall():
      users.append({'username': friend['friend'],
            'profile_picture': friend['profile-picture']})

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

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

```python
    # Get relevant transactions, return rendered "homepage" template
    transactions = []
    cur = connection.execute(
        "SELECT * "
        "FROM transactions "
        "WHERE send_username == ? OR recieve_username == ? ",
        (logname, logname, )     # cur = 내가 send 나 recieve인 transactions
    )
    resp = cur.fetchall()
    for transaction in resp:
      cur_transaction = {"send_username": transaction['send_username'],
          "recieve_username": transaction['recieve_username'],
          transaction_amout          "                         }
      transactions.append(cur_transaction)     # transactions[]

    for cur_friend in users:
      cur = connection.execute(
        "SELECT send_username, recieve, amout, id "
        "FROM transactions "
         WHERE is_public == 1       # cur = transactions중에 public이고 친구의 send 나 recieve
         AND (send_username == ? OR recieve_username == ?) ",
         (cur_friend['username'] ,  "  ,)
      )
      resp = cur.fetchall()       # → transactions[]
                                  # 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)
```




```html
<body>
<h1>Users</h1>

{% for user in users %}
    <div>
        {{user.username}}
        <img src="{{ url_for('get_upload', filename=user.profile_picture) }}">
    </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>
```

```html
<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>
```

```python
@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 = ""              # 입력받은 값들 variable화
    recieve_user = " "

    if(send_or_request == 'send'):
      send_user = logname
      recieve_user = other_user

    if(send_or_request == 'request'):

    # check if transaction requirements met, add to DB if so
    cur = connection.execute(
      "SELECT * "          # cur = 친구들중 입력한것에 해당 친구가 transaction에 있는지
       FROM friends
       WHERE username1 == ? AND username2 == ?
       UNION
       SELECT *   FROM friends
       (send_user, recieve_user, recieve_user, send_ ,)
    )
    possible_friends = cur.fetchall()
    if(not possible_friends):
      return flask.redirect(flask.url_for("show_index"))
    cur = connection.execute(    # cur = user중에 보내는사람
      "SELECT account_balance  FROM users
       WHERE username == ? , (send_user)
    )
    cur_balance = cur.fetchone()['account_balance']
    if(cur_balance - amount < 0):
      return flask.redirect(flask.url_for("show_index"))

    connection.execute(
      "INSERT INTO transactions (send, recieve, amount, ispublic)"
      "VALUES (?, ?, ?, ?),         # transactions에 추가
      (send, recieve, amout, is public)
    )

    return flask.redirect(flask.url_for("show_index"))
```

```python
@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 = []
    resp = cur.fetchall()
    for good in resp:
        cur_good = {'title': good['title'],
                    'id': good['id'],
                    'description': good['des']}
        goods.append(cur_good)
    return flask.jsonify(goods)
```

```python
@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 (?, ?, ?, ?, ?), "        # 입력받은것 Insert.
       (title, description, seller_username, price, 1)
    )
    cur = connection.execute("SELECT MAX(id) as id FROM goods")
    id_dict = cur.fetchone()  or  flask.abort(404)
    cur = connection.execute(       # 그 good (max id)를 goods에서 찾고
       "SELECT * FROM goods WHERE id = ? ",
        (id, )
    )
    query_result = cur.fetchone()
    if(query_result is None):
           flask.abort(404)
    query_result['resource_url'] = f"/api/good/{id}/"     # url 설정
    return flask.jsonify(query_result)
```

```python
@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 == ?~~
        "DELETE FROM goods WHERE id == ? ",
        (good_id, )
    )
```

```javascript
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)     // ↳ common way to send JSON data to a server.
    .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>
}
```