CISC 5550 Cloud Computing Final **Annette Chiu**

## What Is Cloud Computing?

The “cloud” has always been a metaphor for the Internet; in fact, cloud symbols are often used to portray the Internet on diagrams. As a virtual space that connects users from all over the globe, the Internet is like a cloud, sharing information by way of satellite networks.

### **Sharing and Storing Data**

Cloud computing, in turn, refers to sharing resources, software, and information via a network, in this case the Internet. The information is stored on physical servers maintained and controlled by a cloud computing provider, such as Apple in regards to [iCloud](https://www.icloud.com/). As a user, you access your stored information on the cloud via the Internet. By using cloud storage, you don’t have to store the information on your own hard drive. Instead, you can access it from any location and download it onto any device of your choice, including [laptops](https://www.moneycrashers.com/best-budget-laptop-computers/), [tablets](https://www.moneycrashers.com/ipad-alternatives-cheap-tablets/), or [smartphones](https://www.moneycrashers.com/best-cell-phones/). Moreover, you can also edit files, such as Word documents or PowerPoint presentations, simultaneously with other users, making it easier to work away from the office. There are different types of cloud computing services available to suit different needs. While some cater to individual users who want to store photos, documents, and videos, others are destined for companies that need extensive platforms to develop IT applications, for example. Depending on your needs, the prices will vary. As an individual user, you can get an initial amount of storage for free, such as 5GB with iCloud. If you need additional storage, you will have to pay a fee. Fees are usually set at monthly or yearly rates, depending on the services you are using.

**Cloud Development and Deployment Project**

The files in the files is : 1 ) Login.html 2 ) testDb.py 3) todolist.py 4) index.html 5) todolist\_api.py

1. Using vue.js, rewritten the database, modified the API
2. User login is a user who does not go abroad. Create a user directly. If the user is wrong, an error will be reported.
3. When the user logs in, the user does not appear, the system directly creates a new user by default. he existing user will report an error if the password is wrong

Login.html

<!DOCTYPE html>  
<html lang="en">  
  
<head>  
 <title>Todo List Example</title>  
 <meta charset="utf-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1">  
 <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">  
 <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>  
 <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>  
 <script src="https://cdn.staticfile.org/vue/2.2.2/vue.min.js"></script>  
 <style>  
 .done {  
 text-decoration: line-through;  
 }  
 </style>  
</head>  
  
<body>  
 <h2>Please login with User Name and Password </h2>  
 <h2>\_\_\_\_\_</h2>  
 <div class="container">  
 <form action="/login" method="POST" id="add-form">  
 <div class="row">  
 <div class="col-sm-6">  
 User Name:  
 <input type="text" size="50" name="name" value="" />  
 </div>  
 <div class="col-sm-3">  
 Password:  
 <input type="text" name="pw" value="" />  
 </div>  
 <div class="col-sm-3">  
 <input type="submit" value="login" />  
 </div>  
 </div>  
 </form>  
 </div>  
 <div id="app">  
 <template v-if="error">  
 <h1>Ｕser name or password is wrong, please try again!</h1>  
 </template>  
 </div>  
  
 <script>  
 new Vue({  
 el: '#app',  
 data: {  
 {% if errorInfo %} error: true {% else %} error: false {% endif %}  
 }  
 })  
 </script>  
</body>  
  
</html>

testDb.py

**import** sqlite3  
  
sqlite\_db = sqlite3.connect("todolist.db")  
# res = sqlite\_db.execute("""CREATE TABLE users (  
# ID INTEGER PRIMARY KEY AUTOINCREMENT,  
# NAME TEXT NOT NULL,  
# PW TEXT NOT NULL  
# );""")  
#  
# res = sqlite\_db.execute("""CREATE TABLE entries (  
# ID INTEGER NOT NULL,  
# what\_to\_do TEXT NOT NULL,  
# due\_date TEXT NOT NULL,  
# status TEXT NOT NULL  
# );""")  
#  
#  
#  
# # res = sqlite\_db.execute("SELECT name FROM sqlite\_master WHERE type='table';")  
# res = sqlite\_db.execute("""INSERT INTO users (NAME, PW)  
# values ('test1', '1');""")  
# res = sqlite\_db.execute("""INSERT INTO users (NAME, PW)  
# values ('test2', '2');""")  
  
# res = sqlite\_db.execute("""select NAME from users where ID = 1;""")  
# sqlite\_db.commit()  
# print(list(res))  
#  
# sqlite\_db.close();  
  
**import** socket  
  
# 查看当前主机名  
**print**('当前主机名称为 : ' + socket.gethostname())  
  
# 根据主机名称获取当前IP  
**print**('当前主机的IP为: ' + socket.gethostbyname(socket.gethostname()))

todolist.py

# RESTful API  
**from** flask **import** Flask, render\_template, redirect, g, request, url\_for, jsonify, Response  
**import** sqlite3  
**import** urllib  
**import** json  
  
DATABASE = 'todolist.db'  
  
  
app = Flask(\_\_name\_\_)  
app.config.from\_object(\_\_name\_\_)  
  
  
@app.route("/api/items/<userid>") # default method is GET  
**def get\_items**(userid):  
 userid = urllib.parse.unquote(userid)  
 db = get\_db()  
 cur = db.execute('SELECT what\_to\_do, due\_date, status FROM entries where ID="'+userid+'"')  
 entries = cur.fetchall()  
 tdlist = [dict(what\_to\_do=row[0], due\_date=row[1], status=row[2])  
 **for** row **in** entries]  
 response = Response(json.dumps(tdlist), mimetype='application/json')  
 **return** response  
  
@app.route("/api/username/<userid>")  
**def get\_username**(userid):  
 userid = urllib.parse.unquote(userid)  
 db = get\_db()  
 res = db.execute('SELECT NAME FROM users where ID="'+userid+'"')  
 res = list(res)  
 **print**(res)  
 **return** jsonify({"name": res[0][0]})  
  
  
@app.route("/api/login", methods=['POST'])  
**def login**():  
 db = get\_db()  
 name = request.json['NAME']  
 pw = request.json['PW']  
 sql\_str = 'select ID from users where NAME="' + name + '";'  
 res = db.execute(sql\_str)  
 res = list(res)  
 **if** len(res) == 0:  
 db.execute("insert into users (NAME, PW) values ('" + name + "','"+pw+"')")  
 sql\_str = 'select ID from users where NAME="' + name + '" and PW="' + pw + '";'  
 # print(sql\_str)  
 res = db.execute(sql\_str)  
 res = list(res)  
 # print(res)  
 db.commit()  
 **if** len(res) == 0:  
 **return** jsonify({"result": False})  
 **else**:  
 **return** jsonify({"result": True, "userid":res[0]})  
  
  
@app.route("/api/items", methods=['POST'])  
**def add\_item**():  
 db = get\_db()  
 db.execute('insert into entries (ID, what\_to\_do, due\_date, status) values (?, ?, ?, ?)',  
 [request.json['ID'], request.json['what\_to\_do'], request.json['due\_date'], 'processing'])  
 db.commit()  
 **return** jsonify({"result": True})  
  
  
@app.route("/api/items/<userid>/<item>", methods=['DELETE'])  
**def delete\_item**(userid, item):  
 userid = urllib.parse.unquote(userid)  
 item = urllib.parse.unquote(item)  
 db = get\_db()  
 db.execute("DELETE FROM entries WHERE what\_to\_do='"+item+"' and ID='" + userid + "'")  
 db.commit()  
 **return** jsonify({"result": True})  
  
  
@app.route("/api/items/<userid>/<item>", methods=['PUT'])  
**def update\_item**(userid, item):  
 # we do not need the body so just ignore it  
 userid = urllib.parse.unquote(userid)  
 item = urllib.parse.unquote(item)  
 db = get\_db()  
 db.execute("UPDATE entries SET status='done' WHERE what\_to\_do='"+item+"' and ID='" + userid + "'")  
 db.commit()  
 **return** jsonify({"result": True})  
  
  
**def get\_db**():  
 *"""Opens a new database connection if there is none yet for the  
 current application context.  
 """* **if not** hasattr(g, 'sqlite\_db'):  
 g.sqlite\_db = sqlite3.connect(app.config['DATABASE'])  
 **return** g.sqlite\_db  
  
  
@app.teardown\_appcontext  
**def close\_db**(error):  
 *"""Closes the database again at the end of the request."""* **if** hasattr(g, 'sqlite\_db'):  
 g.sqlite\_db.close()  
  
  
**if** \_\_name\_\_ == "\_\_main\_\_":  
 app.run("0.0.0.0", port=5001)

