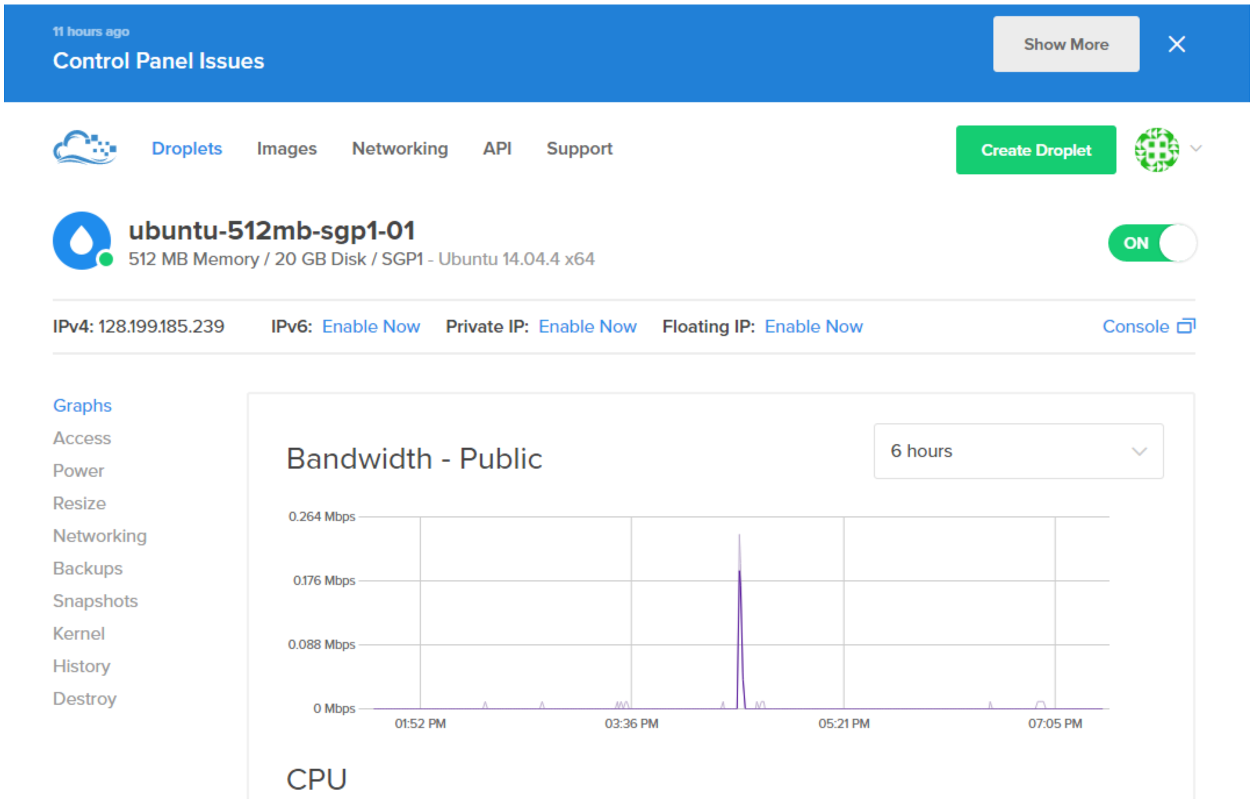
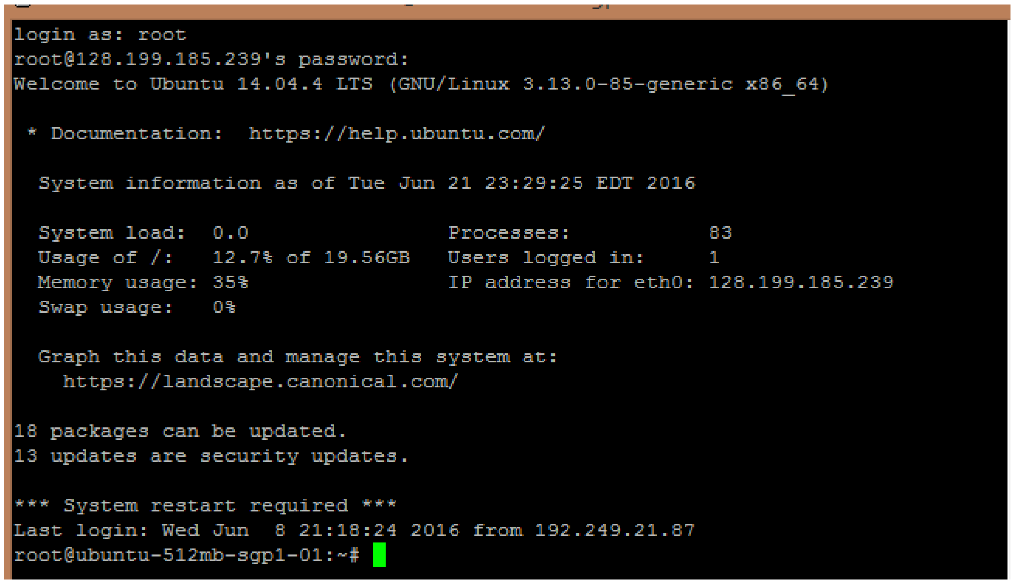
Server:

In order to satisfy the multimedia characteristic of data shared via network, I purchased a server which costs 5 dollars per month, which is located in Singapore. The server is based on Ubuntu and has IP address of 128.199.185.239. After connecting to the server using putty, I installed apache2.



(digital ocean)



```
login as: root
root@128.199.185.239's password:
Welcome to Ubuntu 14.04.4 LTS (GNU/Linux 3.13.0-85-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

  System information as of Tue Jun 21 23:29:25 EDT 2016

  System load:  0.0               Processes:           83
  Usage of /:   12.7% of 19.56GB  Users logged in:     1
  Memory usage: 35%               IP address for eth0: 128.199.185.239
  Swap usage:   0%

  Graph this data and manage this system at:
    https://landscape.canonical.com/

18 packages can be updated.
13 updates are security updates.

*** System restart required ***
Last login: Wed Jun  8 21:18:24 2016 from 192.249.21.87
root@ubuntu-512mb-sgp1-01:~#
```

(Connecting to server using putty)

I made our website using python based programming language called Flask. I configured the Flask by making FlaskApp.conf file. In this file I included the IP adress, wgsi directory, and others.

FlaskApp.conf:

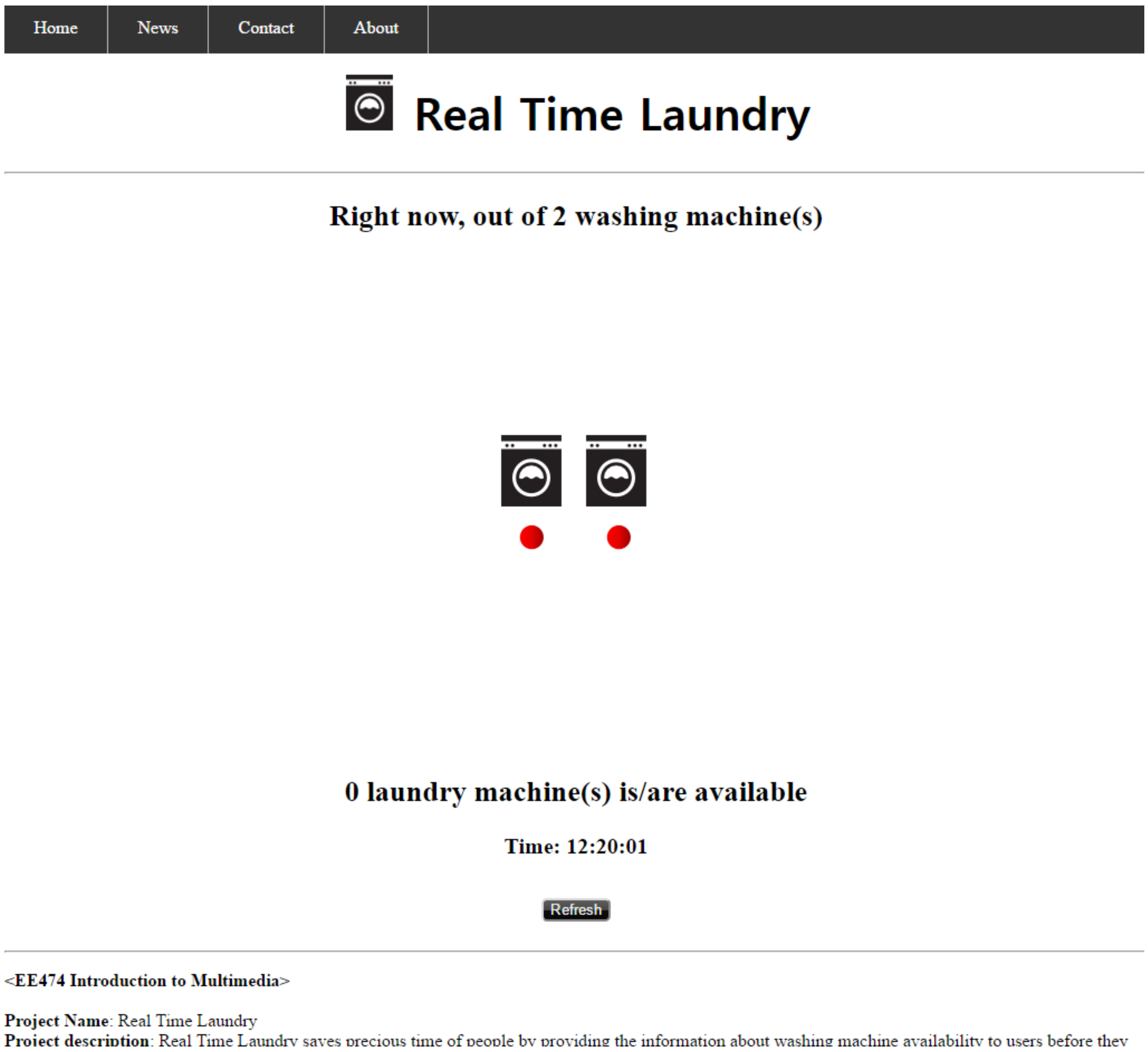
```
<VirtualHost *:80>

        ServerName 128.199.185.239

        ServerAdmin youemail@email.com

        WSGIScriptAlias / /var/www/FlaskApp/flaskapp.wsgi

        <Directory /var/www/FlaskApp/FlaskApp/>

            Order allow,deny

            Allow from all

        </Directory>

        Alias /static /var/www/FlaskApp/FlaskApp/static

        <Directory /var/www/FlaskApp/FlaskApp/static/>

            Order allow,deny

            Allow from all

        </Directory>

        ErrorLog ${APACHE_LOG_DIR}/error.log

        LogLevel warn

        CustomLog ${APACHE_LOG_DIR}/access.log combined

</VirtualHost>
```

Basically what this file does is that, when you enter the servername, 128.199.185.239, on browser, you get the website that I am going to make with Flask program.

After, I made __init__.py, which is the program that controls the website. The source code for __init__.py:

```python
from flask import Flask, render_template

app = Flask(__name__)

@app.route('/')
def homepage():

    with app.open_resource('count.txt') as f:
        laundry_count = f.read()
    with app.open_resource('ava.txt') as f2:
        ava = f2.read()

    with app.open_resource('left.txt') as f3:
        left = f3.read()

    with app.open_resource('right.txt') as f4:
        right = f4.read()

    return render_template('index.html', laundry_count = laundry_count, num =
ava, left_stat = left, right_stat = right)


@app.route('/index')
def index():
    with app.open_resource('count.txt') as f:
        laundry_count = f.read()

    with app.open_resource('ava.txt') as f2:
        ava = f2.read()

    with app.open_resource('left.txt') as f3:
        left = f3.read()

    with app.open_resource('right.txt') as f4:
        right = f4.read()

    return render_template('index.html', laundry_count = laundry_count, num =
ava, left_stat = left, right_stat = right)

@app.route('/news')
def news():
    return render_template('news.html')
    return('hello')
@app.route('/contact')
def contact():
    return render_template('contact.html')

@app.route('/about')
def about():
    return render_template('about.html')

if __name__ == "__main__":
    app.run()
```

In this python code, @app.route() function deals with directories. So for example, if we type http://128.199.185.239/index, the code under @app.route('/index') is executed. When http://128.199.185.239/news is typed, the code under @app.route(/news') is executed, and so on. So when http://128.199.185.239/news is typed, the function render_template('news.html) is executed. The function render_template renders the html file and displays on the website. As a result, when http://128.199.185.23/news is typed on a browser's link, the file called news.html will be displayed on the website. I have four html files for our website: index.html, news.html, contact.html, and about.html. The four html files are shown below:


Home News Contact About

Real Time Laundry

Right now, out of 2 washing machine(s)

0 laundry machine(s) is/are available

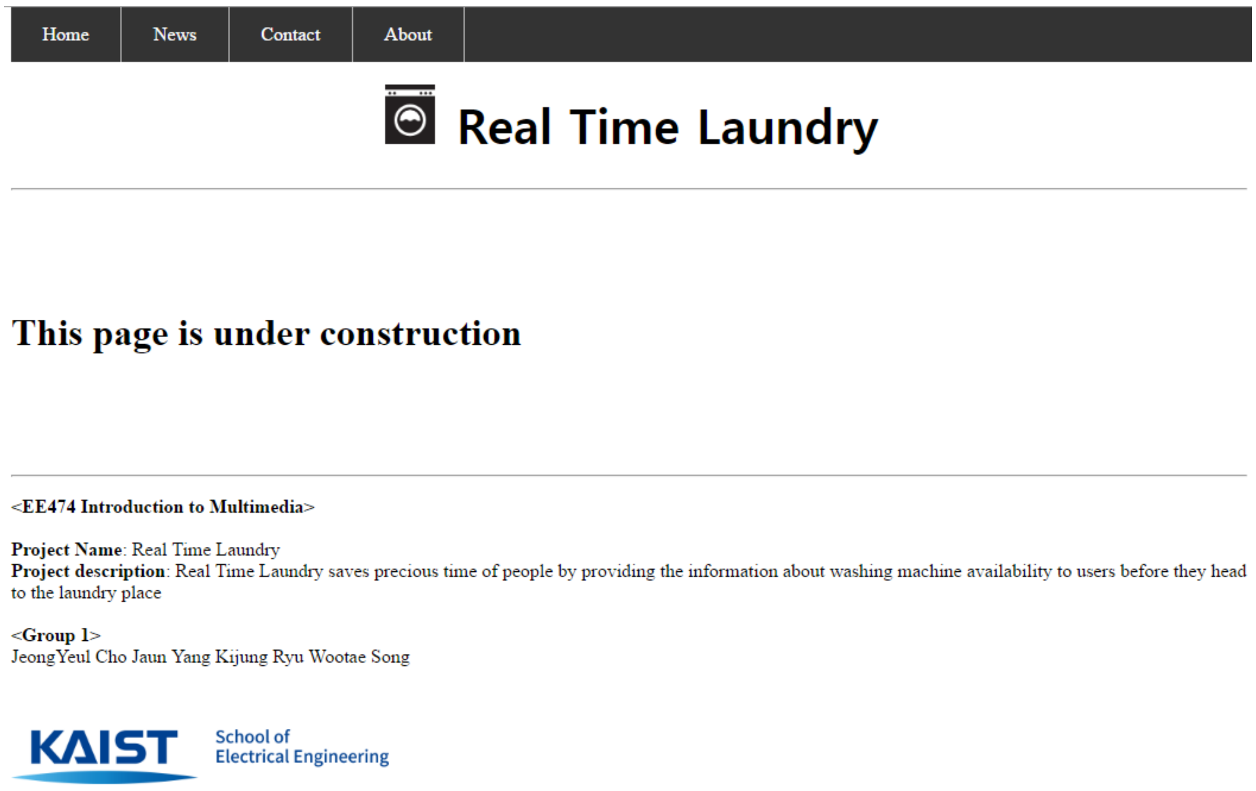
Time: 12:20:01

Refresh

<EE474 Introduction to Multimedia>

Project Name: Real Time Laundry
Project description: Real Time Laundry saves precious time of people by providing the information about washing machine availability to users before they

(http://128.199.185.239/index) (index.html)



Real Time Laundry

This page is under construction

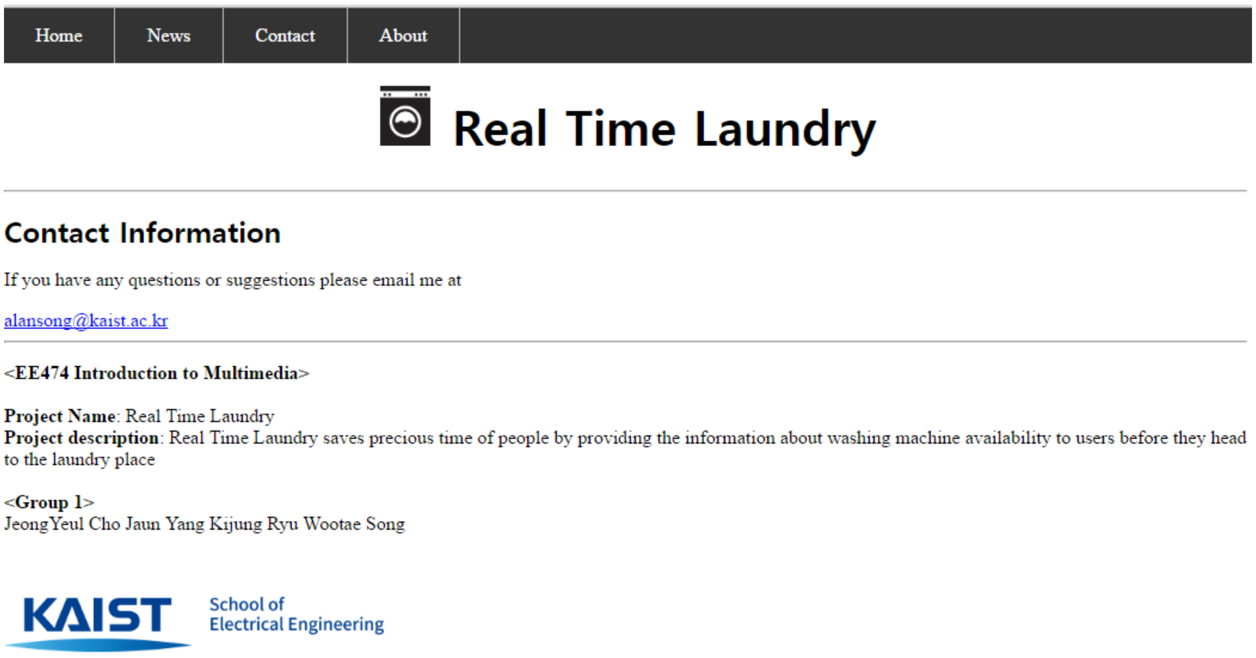
<EE474 Introduction to Multimedia>

Project Name: Real Time Laundry
Project description: Real Time Laundry saves precious time of people by providing the information about washing machine availability to users before they head to the laundry place

<**Group 1**>
JeongYeul Cho Jaun Yang Kijung Ryu Wootae Song

KAIST School of Electrical Engineering

(http://128.199.185.239/news) (news.html)



Real Time Laundry

Contact Information

If you have any questions or suggestions please email me at

alansong@kaist.ac.kr

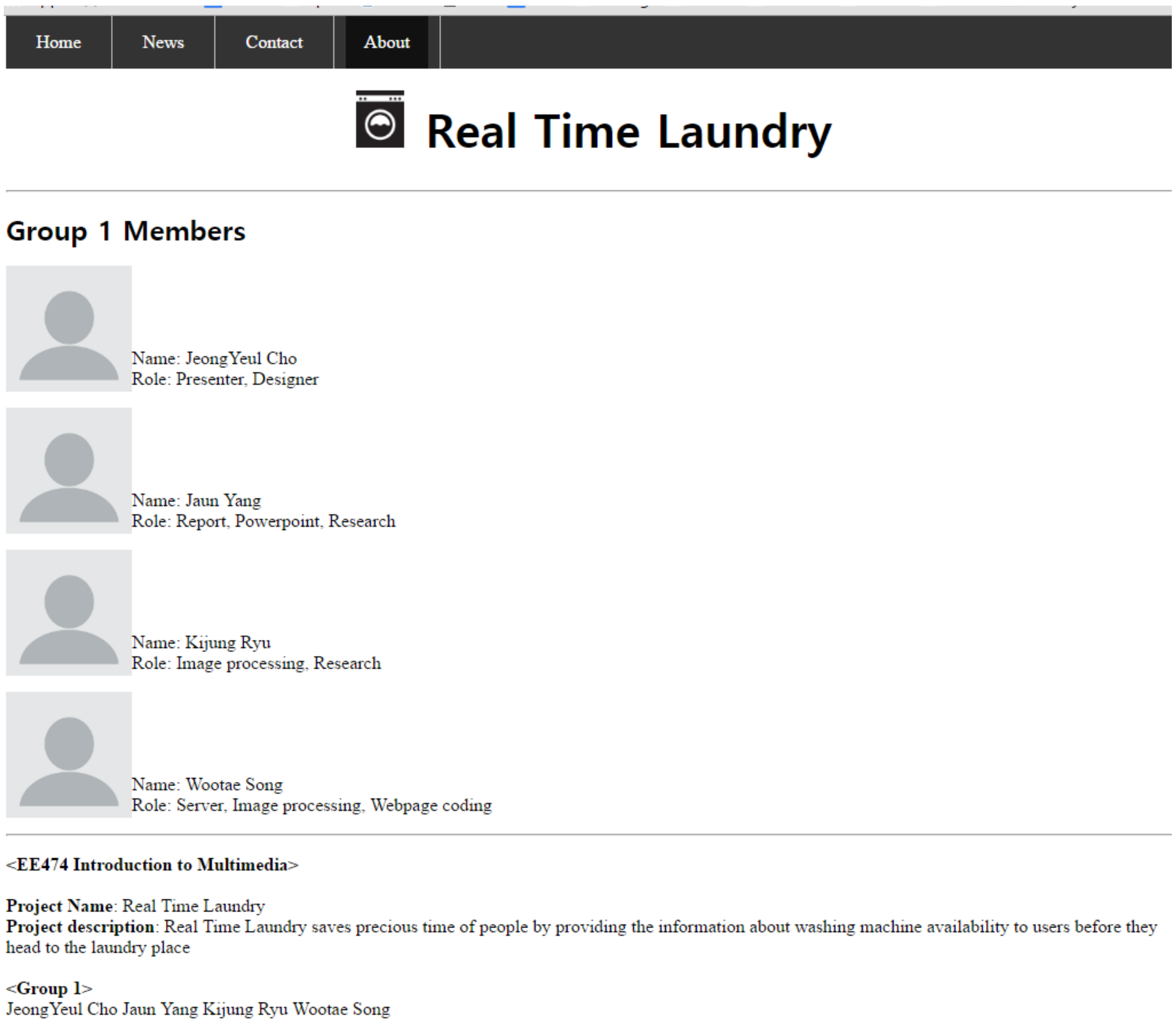
<EE474 Introduction to Multimedia>

Project Name: Real Time Laundry
Project description: Real Time Laundry saves precious time of people by providing the information about washing machine availability to users before they head to the laundry place

<**Group 1**>
JeongYeul Cho Jaun Yang Kijung Ryu Wootae Song

KAIST School of Electrical Engineering

(http://128.199.185.239/contact) (contact.html)


Home News Contact About

Real Time Laundry

Group 1 Members

Name: JeongYeul Cho
Role: Presenter, Designer

Name: Jaun Yang
Role: Report, Powerpoint, Research

Name: Kijung Ryu
Role: Image processing, Research

Name: Wootae Song
Role: Server, Image processing, Webpage coding

<**EE474 Introduction to Multimedia**>

Project Name: Real Time Laundry
Project description: Real Time Laundry saves precious time of people by providing the information about washing machine availability to users before they head to the laundry place

<**Group 1**>
JeongYeul Cho Jaun Yang Kijung Ryu Wootae Song

(http://128.199.185.239/about) (about.html)

Index.html is the main page that displays the availability of laundries and the current time. News.html is the page that displays news related to Real Time Laundry. Unfortunately, there is not news about it yet, so this page is under construction. Contact.html is the page that displays the contact information so that clients can contact to administrator if they have any problems or questions. About.html displays the group members and what they did in this project. Obviously, there are many designs and images in each pages. This is done by combination of html and CSS. The theme.css file is the CSS style sheet that decorates our website and the source code is shown below:

```
p#status {

}
```

```
button {
  border: solid 1px somecolor;
  color: #eee;
  border-radius:5px;
 -moz-border-radius:5px;
  background: -moz-linear-gradient(top, #5E5E5E 0%, #474747 51%, #0a0e0a 51%,
#0a0809 100%);
  background: -webkit-gradient(linear, left top, left bottom, color-
stop(0%,#5E5E5E), color-stop(51%,#474747), color-stop(51%,#0a0e0a), color-
stop(100%,#0a0809));
}
button:hover {
  color: white;
}

h2#main {
      font-size: 165%;
}

div.first {
      height: 30px;
}

img#laundry {
      height: 100px;
      width: 100px;
}

img#on {
      height:50px;
      width: 50px;
}

img#off {
      height:50px;
      width: 50px;
}

img#icon {
      height: 50px;
      width: 50px;
}

img#kaist {
      width: 318px;
      height: 46px;
}

img#profile {
      float:left;
      width: 110px;
      height: 110px;
}

span#blue {
      font-weight: bold;
}

h1#firstHeader {
      font-family: 'Malgun Gothic';
      font-size: 250%;
}

h1#about {
      font-family: 'Malgun Gothic';
      font-size: 150%;
}

ul {
    list-style-type: none;
    margin: 0;
    padding: 0;
    overflow: hidden;
    background-color: #333;
    width: 100%;
      position: fixed;
      top: 0;
    width: 100%
}

li {
      text-align: center;
      padding: 0% 1%;
      border-right: 1px solid #bbb;
      display: inline;
      float: left;
}

li:last-child {
    border-bottom: none;
}

li a {
    display: block;
    color: white;
    text-align: center;
    padding: 14px 16px;
    text-decoration: none;
}

li a:hover {
    background-color: #111;
}

.active {
    background-color: #4CAF50;
    color: white;
}
```

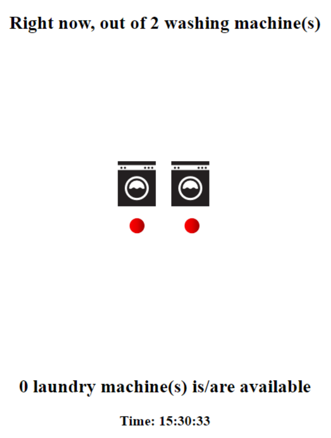
This file assigns text font, text color, text size, image size, back ground color, and so on.

As you can see in our webpage, news.html, contact.html, and about.html are static that is their contents do not change. However, the index.html changes constantly whenever there is an input. The purpose of index.html is to show the availability of laundry machines at the moment. Therefore, index.html should be able to display how many laundry machines are available at the moment dynamically. Html files are static and they cannot display webpage dynamically. Therefore, I added Javascript to index.html to display the webpage dynamically. The source code of index.html is shown below:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<title> Real Time Laundry </title>
<link rel="icon" href= {{ url_for('static', filename='img/icon.png') }} >
<link rel="stylesheet" type="text/css" href=" {{ url_for('static',
filename='css/theme.css') }} ">

<!-- timer -->
<script>
function startTime() {
    var today = new Date();
    var h = today.getHours();
    var m = today.getMinutes();
    var s = today.getSeconds();
    m = checkTime(m);
    s = checkTime(s);
    document.getElementById('txt').innerHTML =
    h + ":" + m + ":" + s;
    var t = setTimeout(startTime, 500);
}
function checkTime(i) {
    if (i < 10) {i = "0" + i};  // add zero in front of numbers < 10
    return i;
}
</script>


</head>
<body onload="startTime()">

<ul>
  <li><a href="/index">Home</a></li>
  <li><a href="/news">News</a></li>
  <li><a href="/contact">Contact</a></li>
  <li><a href="/about">About</a></li>
</ul>

<div class="first"> </div>

<center> <h1 id="firstHeader"> <img id="icon" src= "{{ url_for('static',
filename='img/icon.png') }} ">  Real Time Laundry  </h1> </center>
<hr>
```

```
<h2 id="main"> <center> Right now, out of {{ laundry_count }} washing
machine(s) </center> </h2>

<br><br><br><br><br><br><br><br>

<center> <p id="status"> </p> </center>

<script>
var count = '{{ laundry_count }}';
var left = '{{ left_stat }}';
var right = '{{ right_stat }}';

var img;

if (count == "1") {
      if (left == "1") {
            img = "<img id=\"laund\" src= \"{{ url_for('static',
filename='img/icon.png') }} \"> <br> <img id=\"on\" src=
\"{{ url_for('static', filename='img/on.png') }} \">";
      } else {
            img = "<img id=\"laund\" src= \"{{ url_for('static',
filename='img/icon.png') }} \"> <br> <img id=\"off\" src=
\"{{ url_for('static', filename='img/off.png') }} \">";
      }
}

if (count == "2") {
      if (left == "0" && right == "1") {
            img = "<img id=\"laund\" src= \"{{ url_for('static',
filename='img/icon.png') }} \"> &nbsp; <img id=\"laund\" src=
\"{{ url_for('static', filename='img/icon.png') }} \"> <br> <img id=\"off\"
src= \"{{ url_for('static', filename='img/off.png') }} \"> &nbsp; &nbsp;
&nbsp; <img id=\"on\" src= \"{{ url_for('static', filename='img/on.png') }}
\">";
      } else if (left == "1" && right == "0") {
            img = "<img id=\"laund\" src= \"{{ url_for('static',
filename='img/icon.png') }} \"> &nbsp; <img id=\"laund\" src=
\"{{ url_for('static', filename='img/icon.png') }} \"> <br> <img id=\"on\"
src= \"{{ url_for('static', filename='img/on.png') }} \"> &nbsp; &nbsp;
&nbsp; <img id=\"off\" src= \"{{ url_for('static', filename='img/off.png') }}
\">";
      } else if (left == "1" && right == "1") {
            img = "<img id=\"laund\" src= \"{{ url_for('static',
filename='img/icon.png') }} \"> &nbsp; <img id=\"laund\" src=
\"{{ url_for('static', filename='img/icon.png') }} \"> <br> <img id=\"on\"
src= \"{{ url_for('static', filename='img/on.png') }} \"> &nbsp; &nbsp;
&nbsp; <img id=\"on\" src= \"{{ url_for('static', filename='img/on.png') }}
\">";
      } else {
            img = "<img id=\"laund\" src= \"{{ url_for('static',
filename='img/icon.png') }} \"> &nbsp; <img id=\"laund\" src=
\"{{ url_for('static', filename='img/icon.png') }} \"> <br> <img id=\"off\"
src= \"{{ url_for('static', filename='img/off.png') }} \"> &nbsp; &nbsp;
&nbsp; <img id=\"off\" src= \"{{ url_for('static', filename='img/off.png') }}
\">";
      }
}

document.getElementById('status').innerHTML = img;


</script>

<br><br><br><br><br><br><br><br>

<h2 id="main"> <center> {{ num }} laundry machine(s) is/are available
</center> </h2>


<div align="center"> <h2 style="font-size:125%"> Time: <span align="center"
id="txt"> </span> </h2> </div>

<br>
<center>
<form method="get" action="/index">
    <button type="submit"> Refresh </button>
</form>
</center>
<br>

<hr>
<p id="end"> <span id="blue"> &lt;EE474 Introduction to Multimedia&gt;
</span> <br> <br> <span style="font-weight:bold"> Project Name</span>: Real
Time Laundry <br> <span style="font-weight:bold">Project description</span>:
Real Time Laundry saves precious time of people by providing
the information about washing machine availability to users before they head
to the laundry place <br> <span id="blue"> <br> &lt;Group 1&gt; </span> <br>
JeongYeul Cho  Jaun Yang  Kijung Ryu  Wootae Song </p>

<br> <br>

<img id="kaist" src= "{{ url_for('static', filename='img/kaist.png') }}">

</body>
</html>
```

There are two things that are changing dynamically in index.html: timer that displays the current time and the information about laundry availability. I used open source for timer but made own Javascript code for the information about laundry availability, which is the most important thing in our webpage.


Right now, out of 2 washing machine(s)

0 laundry machine(s) is/are available

Time: 15:30:33

(the dynamic part in the webpage)

The Javascript code for the information about laundry availability is enclosed in <script> tag. There are three variables in the Javascript code: var count, left, and right. This variable is provided by the __init__.py code. If you look at __init__.py under @app.route('/index'), it uses the function app.open_reousrce() to open text files. After opening the text files, it saves the contents of the text file in variables laundry_count, ava, left, and right. Then, when it uses the render_template function to display the webpage, it also gives the html file the variables that it read with the app.open_resource as parameters. As a result, index.html gets the variables from __init__.py. After, obtaining the variables from __init__.py, it uses if statements to evaluate the variables. It first checks how many laundry machines there are with count variable. If count is 1, it displays one laundry icon; if count is two, it displays two laundry icons. After, it checks left and right variables. If the count is 1, it only checks for left variable; if the count is 2, it checks both left and right variables. Left and right variables contain information about whether a laundry machine is running or not. If left variable is 1, it indicates that left laundry machine is on; if the right variable is 0, it indicates the right laundry machine is off. According to the on/off information, it displays red circle below the corresponding machine if the machine is on and using; it displays green circle below the corresponding machine if the machine is off and available. This information is saved in variable img and displayed in the middle of webpage. This Javascript runs every time a client access the index.html. As a result, to get a new information, a client need refresh the page. Every other codes in the index.html and codes in news.html, contact.html, and about.html are straight forward. There are just text and images.

To summarize, index.html displays the information about the availability of laundry machines by getting variables from __init__.py file, which gets the variables from text files. How are the text files are formed then? There is a program, listener.c that runs continuously in the server. This program is a UDP server program that receives the information about laundry information from MATLAB. The source code for listener.c is shown below:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
```

```c
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>

#define MYPORT "4950"    // the port users will be connecting to

#define MAXBUFLEN 100

FILE *status;
FILE *status2;
FILE *status3;
FILE *status4;

// get sockaddr, IPv4 or IPv6:
void *get_in_addr(struct sockaddr *sa)
{
    if (sa->sa_family == AF_INET) {
        return &(((struct sockaddr_in*)sa)->sin_addr);
    }

    return &(((struct sockaddr_in6*)sa)->sin6_addr);
}

int main(void)
{
    int sockfd;
    struct addrinfo hints, *servinfo, *p;
    int rv;
    int numbytes;
    struct sockaddr_storage their_addr;
    char buf[MAXBUFLEN];
    socklen_t addr_len;
    char s[INET6_ADDRSTRLEN];

    while(1) {
        memset(&hints, 0, sizeof hints);
        hints.ai_family = AF_UNSPEC; // set to AF_INET to force IPv4
        hints.ai_socktype = SOCK_DGRAM;
        hints.ai_flags = AI_PASSIVE; // use my IP

        if ((rv = getaddrinfo(NULL, MYPORT, &hints, &servinfo)) != 0) {
            fprintf(stderr, "getaddrinfo: %s\n", gai_strerror(rv));
            return 1;
        }

        // loop through all the results and bind to the first we can
        for(p = servinfo; p != NULL; p = p->ai_next) {
            if ((sockfd = socket(p->ai_family, p->ai_socktype,
                    p->ai_protocol)) == -1) {
                perror("listener: socket");
                continue;
            }

            if (bind(sockfd, p->ai_addr, p->ai_addrlen) == -1) {
                close(sockfd);
                perror("listener: bind");
                continue;
            }

            break;
        }

        if (p == NULL) {
            fprintf(stderr, "listener: failed to bind socket\n");
            return 2;
        }

        freeaddrinfo(servinfo);

        printf("listener: waiting to recvfrom...\n");

        addr_len = sizeof their_addr;
        if ((numbytes = recvfrom(sockfd, buf, MAXBUFLEN-1 , 0,
            (struct sockaddr *)&their_addr, &addr_len)) == -1) {
            perror("recvfrom");
            exit(1);
        }




        printf("listener: got packet from %s\n",
            inet_ntop(their_addr.ss_family,
                get_in_addr((struct sockaddr *)&their_addr),
                s, sizeof s));
        printf("listener: packet is %d bytes long\n", numbytes);
        buf[numbytes] = '\0';
        printf("listener: packet contains \"%s\"\n", buf);


        char *count = malloc(sizeof(char));
        char *left = malloc(sizeof(char));
        char *right = malloc(sizeof(char));

        count = strtok(buf, " ");


        if (strcmp(count, "1") == 0) {
            left = strtok(NULL, " ");
        } else if (strcmp(count, "2") == 0) {
            left = strtok(NULL, " ");
            right = strtok(NULL, " ");
        }


        status = fopen("count.txt", "w");
        if (status == NULL) {
            fprintf(stderr, "Could not open status.txt\n");
        }
        memset(status, 0, sizeof status);
        fwrite(count, 1, 1, status);


        status2 = fopen("left.txt", "w");
```

```c
if (status2 == NULL) {
    fprintf(stderr, "Could not open status2.txt\n");
}
memset(status2, 0, sizeof status2);
fwrite(left, 1, 1, status2);



if (strcmp(count, "2") == 0) {
    status3 = fopen("right.txt", "w");
    if (status3 == NULL) {
        fprintf(stderr, "Could not open status3.txt\n");
    }
    memset(status3, 0, sizeof status3);
    fwrite(right, 1, 1, status3);
}

char ava[1];


if (strcmp(count, "1") == 0) {
    if (strcmp(left, "1") == 0) {
        ava[0] = '0';
    } else {
        ava[0] = '1';
    }
} else {
    if (strcmp(left, "1") == 0 && strcmp(right, "0") == 0) {
        ava[0] = '1';
    } else if (strcmp(left, "0") == 0 && strcmp(right, "1") == 0 ) {
        ava[0] = '1';
    } else if (strcmp(left, "1") == 0 && strcmp(right, "1") == 0) {
        ava[0] = '0';
    } else {
        ava[0] = '2';
    }
}


status4 = fopen("ava.txt", "w");
if (status4 == NULL) {
    fprintf(stderr, "Could not open status4.txt\n");
}
memset(status4, 0, sizeof status4);
fwrite(ava, 1, 1, status4);



fclose(status);
fclose(status2);

if (strcmp(count, "2") == 0) {
    fclose(status3);
}
fclose(status4);
```

```
        close(sockfd);
    }


    return 0;
}
```

This program creates a UDP socket and binds the port number of 4950 to it. As a result, MATLAB program can send the UDP packets to this server with destination IP address of 128.199.185.239 and port number 4950. After receiving the information about the availability of the laundry machines, it parses the data and saves the data into four textfiles: count.txt, left.txt, right.txt, and ava.txt. These are the text files that __init.py__ reads and sends to index.html.

To summarize, there is a UDP based server program called listener.c that is running all the time. This program receives the information about laundry availability from MATLAB. After, it parses the information and saves the information in four different text files. Then, the __init__.py Flask program reads these text file and sends the information to index.html file. In index.html, it receives the information and uses Javascript to dynamically display the information about laundry availability.

The hierarchy of the server program is shown below:

```
static
    css
        theme.css
    img
        icon.png
        kaist.png
        off.png
        on.png
        profile.png
templates
        about.html
        contact.html
        index.html
        news.html
```

__init__.py

listener.c