

Objective

Create Register and User Detail pages using PUG template engine.

When a user registers store their cookie and retrieve their information on click of profile button. Use `express.static()` to load CCS files for page formatting.

Home Page

The home page contains a register form where the user can input their email and password. On click of 'register', their cookie is stored in `express (res.cookie('cookie', req.body).render('home'))`. The home page is rendered back to the user at this point:

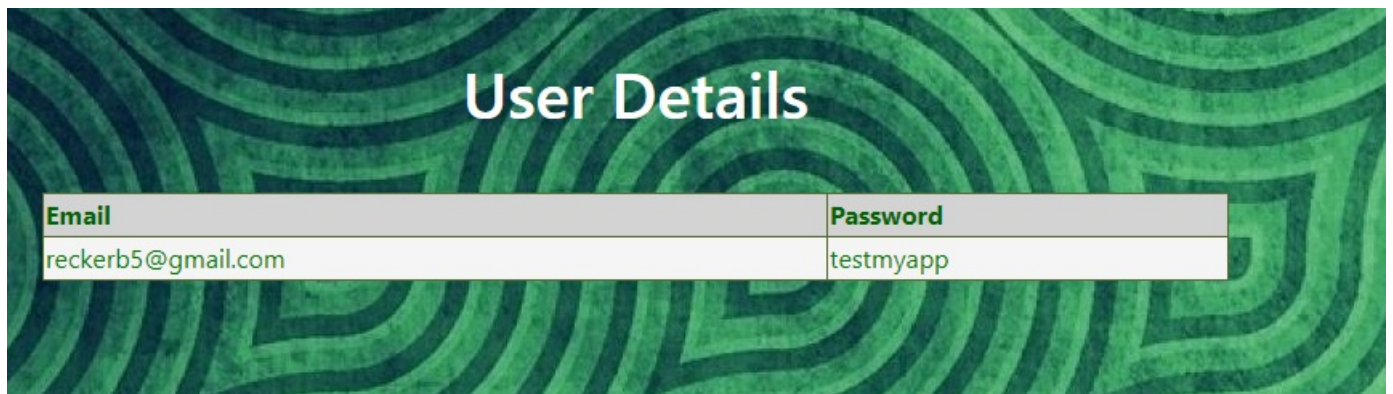


The screenshot shows a web page titled "Lab 9" with a green and black concentric circle pattern background. It features a registration form with the following elements:

- Email Address:** A text input field containing the email address "reckerb5@gmail.com".
- Password:** A password input field with 10 dots representing the masked password.
- Register:** A blue button located below the password field.
- Profile:** A grey button located below the Register button.

```
C:\Users\br030614\Documents\CS5551\CS5551_ASE\lab9>node index.js
Listening to port 3000
email =reckerb5@gmail.com
password =testmyapp
□
```

Then, the user can click the 'profile' button which gets the cookie stored in express and renders the user.pug view to display their details:



User Details

Email	Password
reckerb5@gmail.com	testmyapp

The cookie that is passed can be seen in the console:

```
C:\Users\br030614\Documents\CS5551\CS5551_ASE\lab9>node index.js
Listening to port 3000
email =reckerb5@gmail.com
password =testmyapp
cookie = [object Object]
□
```

Below shows another test of this application for a new user:

Lab 9

Email Address:

test

Password:

....

Register

Profile

User Details

Email	Password
test	test

This application contains the home.pug and user.pug views:

```

html
  head
    title Lab 9
    link(rel="stylesheet" href="/css/styles.css")
    script(src="/js/script.js")
    link(rel='stylesheet', href='/css/bootstrap.min.css')

  body
    form(action='/', method='POST')
      .panel.primary-panel
        .panel-heading
          h2.panel-title Lab 9
        .panel-body
          .form-group
            label Email Address:
            input.form-control(type='text', required, name='email', placeholder='Email')
          .panel.primary-panel
            .form-group
              label Password:
              input.form-control(type='password', name='password', placeholder='password')
          .panel.primary-panel
            button.btn.btn-primary(type='submit') Register

    form.form(method='POST' action='/user')
      button.btn.btn-secondary(type='submit') Profile

```

```

doctype html
html
  head
    title User
    link(rel="stylesheet" href="/css/styles.css")
    script(src="/js/script.js")
    link(rel='stylesheet', href='/css/bootstrap.min.css')

  body
    h1 User Details
    table
      tr
        th Email
        th Password
      tr
        td #{details.email}
        td #{details.password}

```

styles.css and bootstrap.min.css are the stylesheets referenced in the PUG files which are located in the public folder. The public folder is

loaded by express as well (`app.use(express.static('public'));`). The background image is also loaded from the public folder:

```

1  body{
2      background-color: gray;
3      background-image: url('/assets/background.jpg');
4  }
5
6  .primary-panel{
7      width:40%;
8      margin-left: 30%;
9      margin-right: 30%;
10 }
11
12 .panel-heading {
13     margin-top:10px;
14     color: white;
15     text-align: center;
16     padding: 25px;
17 }
18
19 form {
20     font-family: Verdana;
21 }
22
23 .form {
24     text-align: left;
25     padding: 7px;
26     margin-left: 30%;
27 }
28
29 .form-group{
30     color: white;
31     font-weight: 10pt;
32 }
33
34 h1 {
35     padding: 30px;
36     text-align: center;
37     color: white;
38 }
39
40 table, th, td {
41     border: 1px solid darkolivegreen;
42 }
43
44 th {
45     background-color: lightgray;
46     color: darkgreen;
47 }
48
49 t_ {
50     background-color: whitesmoke;

```

tr

Outcome

I found the pug views very easy to stand up along with the CSS. It wasn't too difficult to set up the routes and render the views as expected. I did struggle a little with the cookie portion as I was initially overthinking the process. I could easily set and get the cookie but was having a hard time determining how to pass it to the user.pug view. Once I figured out that it could be added to the render command in express (*res.render('user', {details: req.cookies.cookie});*) I was able to finish my lab.