

**1. Create User table with user with email,username,roll number, password.**

New table ×

RZN80672

User

Add column +

Name	Data type	Nullable	Length	Scale
email	VARCHAR	Y	100	--
username	VARCHAR	Y	100	--
rollno	INT	Y	--	--
password	VARCHAR	Y	100	--

Generate DDL

Create

**INSERT QUERY :**

INSERT INTO User

VALUES ('knarmathait@gmail.com','knarmatha',1001,'knarmatha1001');

INSERT INTO User

VALUES ('harshu@gmail.com','pharshu',1002,'pharshu1002');

INSERT INTO User

VALUES ('vadhani@gmai.com','vadhani',1003,'vadhani1003');

```

1 INSERT INTO User
2 VALUES ('knarmathait@gmail.com','knarmatha',1001,'knarmatha1001');
3 INSERT INTO User
4 VALUES ('harshu@gmail.com','pharshu',1002,'pharshu1002');
5 INSERT INTO User
6 VALUES ('vadhani@gmail.com','vadhani',1003,'vadhani1003');

```

Script	Date	Status	Runtime
^ Untitled - 1	Oct 17, 2022 6:40:42 PM	✓ 2	0.040 s
INSERT INTO User VALUES ('harshu@gmail.com','pharshu',1002,'pharshu1002')		✓	0.020 s
INSERT INTO User VALUES ('vadhani@gmail.com','vadhani',1003,'vadhani1003')		✓	0.020 s
^ Untitled - 1	Oct 17, 2022 6:35:15 PM	✓ 1	0.020 s
INSERT INTO User VALUES ('knarmathait@gmail.com','knarmatha',1001,'knarmatha...		✓	0.020 s

RZN80672.USER				Back
				Export to CSV
EMAIL	↑↓	USERNAME	ROLLNO	PASSWORD
harshu@gmail.com		pharshu	1002	pharshu1002
knarmathait@gmail.com		knarmatha	1001	knarmatha1001
vadhani@gmail.com		vadhani	1003	vadhani1003

## 2. Perform UPDATE,DELETE Queries with user table

### UPDATE:

```

1 UPDATE User
2 SET email = 'narmathak@gmail.com'
3 WHERE rollno = 1001;

```

^ Untitled - 1	Oct 17, 2022 6:53:20 PM	✓ 1	0.020 s
UPDATE User SET email = 'narmathak@gmail.com' WHERE rollno = 1001		✓	0.020 s

EMAIL	USERNAME	ROLLNO	PASSWORD
harshu@gmail.com	pharshu	1002	pharshu1002
narmathak@gmail.com	knarmatha	1001	knarmatha1001
vadhani@gmail.com	vadhani	1003	vadhani1003

**DELETE:**

The screenshot shows a SQL editor with a toolbar at the top containing icons for saving, undo, redo, toggling SQL mode, toggling full screen, deleting, and searching. Below the toolbar, the SQL statement `DELETE FROM User WHERE username='vadhani';` is entered on line 1. Line 2 is currently empty and highlighted in blue.

Script	Date	Status	Runtime
^ Untitled - 1	Oct 17, 2022 6:57:37 PM	✓ 1	0.009 s
DELETE FROM User WHERE username='vadhani'		✓	0.009 s

EMAIL	USERNAME	ROLLNO	PASSWORD
harshu@gmail.com	pharshu	1002	pharshu1002
narmathak@gmail.com	knarmatha	1001	knarmatha1001

**3. Connect python code to db2.**

```
# -*- coding: utf-8 -*-
```

```
"""
```

```
@author: NARMATHA
```

```
"""
```

```
from flask import Flask,render_template,request,redirect,url_for,session
```

```
import ibm_db
```

```
import re
```

```
app=Flask(__name__)
```

```
app.secret_key='a'
```

```
conn=ibm_db.connect("DATABASE=bludb;HOSTNAME=b70af05b-76e4-4bca-a1f5-23dbb4c6a74e.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=32716;SECURITY=SSL;SSLS
erverCertificate=DigiCertGlobalRootCA.crt;UID=rzn80672;PWD=TG48qREyM8hFUOYV",",")
```

```
@app.route('/')
```

```
def home():
```

```
    return render_template('home.html')

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

def login():

    global userid

    msg=""

    if request.method=='POST':

        username = request.form['username']

        password=request.form['password']

        sql="SELECT * FROM USER WHERE USERNAME=? AND PASSWORD=?"

        stmt = ibm_db.prepare(conn,sql)

        ibm_db.bind_param(stmt,1,username)

        ibm_db.bind_param(stmt,2,password)

        ibm_db.execute(stmt)

        account = ibm_db.fetch_assoc(stmt)

        print(account)

        if account:

            session['loggedin']=True

            session['id']=account[username]

            userid=account['USERNAME']

            session['username']=account['USERNAME']

            msg='Logged in successfully'

            return render_template('dashboard.html', msg=msg)

        else:

            msg='Incorrect username/password'
```

```
        return render_template('login.html', msg=msg)

@app.route('/register',method=['GET','POST'])

def register():

    if request.method=='POST':

        username=request.form['username']

        email=request.form['email']

        password=request.form['password']

        sql="SELECT * FROM USER WHERE USERNAME=?"

        stmt = ibm_db.prepare(conn,sql)

        ibm_db.bind_param(stmt,1,username)

        ibm_db.execute(stmt)

        account = ibm_db.fetch_assoc(stmt)

        print(account)

        if account:

            msg="Account already exists"

        elif not re.match(r'^@]+@[^@]+\.[^@]+' ,email):

            msg="Invalid email address"

        elif not re.match(r'[A-Za-z0-9]+' ,username):

            msg="Name must contain characters and numbers"

        else:

            insert_sql="INSERT INTO USER VALUES(?,?,?)"

            prep_stmt=ibm_db.prepare(conn,insert_sql)

            ibm_db.bind_param(prepare_stmt,1,username)

            ibm_db.bind_param(prepare_stmt,2,email)

            ibm_db.bind_param(prepare_stmt,3,password)
```

```
        ibm_db.execute(prepare_stmt)

        msg="You have successfully registered "

    elif request.method=="POST":

        msg="Please fill out the form"

        return render_template('register.html', msg=msg)

@app.route('/dashboard')

def dash():

    return render_template('dashboard.html')

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

def app():

    msg=""

    if request.method=="POST":

        username=request.form['username']

        email=request.form['email']

        qualification=request.form['qualification']

        skills=request.form['skills']

        jobs=request.form['s']

        stmt=ibm_db.prepare(conn,sql)

        ibm_db.bind_param(stmt,1,username)

        ibm_db.execute(stmt)

        account=ibm_db.fetch_assoc(stmt)

        print(account)

        if account():

            msg="There's only 1 job position"

            return render_template('apply.html',msg=msg)
```

```
insert_sql="INSERT INTO job VALUES(?,?,?,?)"

prep_stmt=ibm_db.prepare(conn,insert_sql)

ibm_db.bind_param(prepare_stmt,1,username)

ibm_db.bind_param(prepare_stmt,2,email)

ibm_db.bind_param(prepare_stmt,3,qualification)

ibm_db.bind_param(prepare_stmt,4,skills)

ibm_db.bind_param(prepare_stmt,5,jobs)

ibm_db.execute(prepare_stmt)

msg="You have successfully applied for job"

session['loggedin']=True

TEXT="Hello user, a new application for job position"+jobs+"isrequested"

elif request.method=="POST":

    msg="Please fill out the form"

    return render_template('apply.html', msg=msg)

@app.route('/display')

def display():

    print (session["username"],session['id'])

    cursor=mysql.connection.cursor()

    cursor.execute('SELECT * FROM job WHERE userid=%s',(session['id'],))

    account=cursor.fetchone()

    print("accountdisplay",account)

    return render_template('display.html',account=account)

@app.route('/logout')

def logout():

    session.pop('loggedin',None)
```

```
session.pop('id',None)

session.pop('username',None)

return render_template('home.html')

if __name__=='__main__':

    app.run(host='0.0.0.0')
```

**4. Create a flask app with registration page, login page and welcome page. By default load the registration page once the user enters all the fields store the data in database and navigate to login page authenticate user username and password. If the user is valid show the welcome page**

**app.py**

```
# -*- coding: utf-8 -*-

"""

@author: NARMATHA

"""

from flask import Flask, render_template, request, redirect, url_for, session

import ibm_db

import re

app = Flask(__name__)

app.secret_key = 'a'

conn=ibm_db.connect("DATABASE=bludb;HOSTNAME=b70af05b-76e4-4bca-a1f5-
23dbb4c6a74e.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=32716;SECURITY=SSL;SSLS
erverCertificate=DigiCertGlobalRootCA.crt;UID=rzn80672;PWD=TG48qREyM8hFUOYV","")

@app.route('/')

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

def login():

    msg = "
```



```
if request.method == 'POST' and 'username' in request.form and 'password' in request.form:

    username = request.form['username']

    password = request.form['password']

    cursor=mysql.connection.cursor()

    cursor.execute('SELECT * FROM accounts WHERE username = % s AND password = % s',
(username, password, ))

    account = cursor.fetchone()

    if account:

        session['loggedin'] = True

        session['id'] = account['id']

        session['username'] = account['username']

        msg = 'Logged in successfully !'

        return render_template('index.html', msg = msg)

    else:

        msg = 'Incorrect username / password !'

        return render_template('login.html', msg = msg)

@app.route('/logout')

def logout():

    session.pop('loggedin', None)

    session.pop('id', None)

    session.pop('username', None)

    return redirect(url_for('login'))

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

def register():

    msg = "
```

```
if request.method == 'POST' and 'username' in request.form and 'password' in request.form and  
'email' in request.form :
```

```
    username = request.form['username']
```

```
    password = request.form['password']
```

```
    email = request.form['email']
```

```
    cursor=mysql.connection.cursor()
```

```
    cursor.execute('SELECT * FROM accounts WHERE username = % s', (username, ))
```

```
    account = cursor.fetchone()
```

```
    if account:
```

```
        msg = 'Account already exists !'
```

```
    elif not re.match(r'^@+@[^@]+\.[^@]+', email):
```

```
        msg = 'Invalid email address !'
```

```
    elif not re.match(r'[A-Za-z0-9]+', username):
```

```
        msg = 'Username must contain only characters and numbers !'
```

```
    elif not username or not password or not email:
```

```
        msg = 'Please fill out the form !'
```

```
    else:
```

```
        cursor.execute('INSERT INTO accounts VALUES (NULL, % s, % s, % s)', (username, password,  
email, ))
```

```
        ibm_db.execute()
```

```
        msg = 'You have successfully registered !'
```

```
    elif request.method == 'POST':
```

```
        msg = 'Please fill out the form !'
```

```
    return render_template('register.html', msg = msg)
```

**login.html**

```
<html>

<head>

    <meta charset="UTF-8">

    <title> Login </title>

    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">

</head>

<body></br></br></br></br></br>

    <div align="center">

        <div align="center" class="border">

            <div class="header">

                <h1 class="word">Login</h1>

            </div></br></br></br>

            <h2 class="word">

                <form action="{{ url_for('login') }}" method="post">

                    <div class="msg">{{ msg }}</div>

                    <input id="username" name="username" type="text" placeholder="Enter Your
Username" class="textbox"/></br></br>

                    <input id="password" name="password" type="password" placeholder="Enter Your
Password" class="textbox"/></br></br></br>

                    <input type="submit" class="btn" value="Sign In"></br></br>

                </form>

            </h2>

            <p class="bottom">Don't have an account? <a class="bottom" href="{{url_for('register')}}">
Sign Up here</a></p>

        </div>
```

```
</div>  
</body>  
</html>
```

A login form design on a light blue background. At the top, there is a dark blue rectangular box containing the word "Login" in a white, italicized serif font. Below this, there are three dark blue rounded rectangular input fields. The first two are stacked vertically and contain the text "Enter Your Username" and "Enter Your Password" respectively, both in a white, italicized serif font. The third field is centered below the others and contains the text "Sign In" in a white, italicized serif font. At the bottom of the form, there is a line of text in a dark blue, italicized serif font that reads "Dont't have an account? [Sign Up here](#)".

*Login*

*Enter Your Username*

*Enter Your Password*

*Sign In*

*Dont't have an account? [Sign Up here](#)*

**register.html:**

```
<html>  
  
<head>  
  
  <meta charset="UTF-8">  
  
  <title> Register </title>
```

```
<link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">

</head>

<body></br></br></br></br></br>

<div align="center">

<div align="center" class="border">

<div class="header">

<h1 class="word">Register</h1>

</div></br></br></br>

<h2 class="word">

<form action="{{ url_for('register') }}" method="post">

<div class="msg">{{ msg }}</div>

<input id="username" name="username" type="text" placeholder="Enter Your
Username" class="textbox"/></br></br>

<input id="password" name="password" type="password" placeholder="Enter Your
Password" class="textbox"/></br></br>

<input id="email" name="email" type="text" placeholder="Enter Your Email ID"
class="textbox"/></br></br>

<input type="submit" class="btn" value="Sign Up"></br>

</form>

</h2>

<p class="bottom">Already have an account? <a class="bottom" href="{{url_for('login')}}">
Sign In here</a></p>

</div>

</div>

</body>

</html>
```

Sign In here' is displayed in a dark blue, italicized font."/>

*Register*

*Enter Your Username*

*Enter Your Password*

*Enter Your Email ID*

*Sign Up*

*Already have an account? [Sign In here](#)*

**index.html**

```
<html>
```

```
<head>
```

```
<meta charset="UTF-8">
```

```
<title> Index </title>
```

```
<link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
```

```
</head>
```

```
<body></br></br></br></br></br>
```

```
<div align="center">

<div align="center" class="border">

  <div class="header">

    <h1 class="word">Index</h1>

  </div></br></br></br>

  <h1 class="bottom">

    Hi {{session.username}}!!</br></br> Welcome to the index page...

  </h1></br></br></br>

  <a href="{{ url_for('logout') }}" class="btn">Logout</a>

</div>

</div>

</body>

</html>
```

**style.css:**

```
.header{

  padding: 5px 120px;

  width: 150px;

  height: 70px;

  background-color: #236B8E;

}

.border{

  padding: 80px 50px;

  width: 400px;
```

```
height: 450px;  
  
border: 1px solid #236B8E;  
  
border-radius: 0px;  
  
background-color: #9AC0CD;  
  
}
```

```
.btn {  
  
padding: 10px 40px;  
  
background-color: #236B8E;  
  
color: #FFFFFF;  
  
font-style: oblique;  
  
font-weight: bold;  
  
border-radius: 10px;  
  
}
```

```
.textbox{  
  
padding: 10px 40px;  
  
background-color: #236B8E;  
  
text-color: #FFFFFF;  
  
border-radius: 10px;  
  
}
```

```
::placeholder {  
  
color: #FFFFFF;  
  
opacity: 1;
```

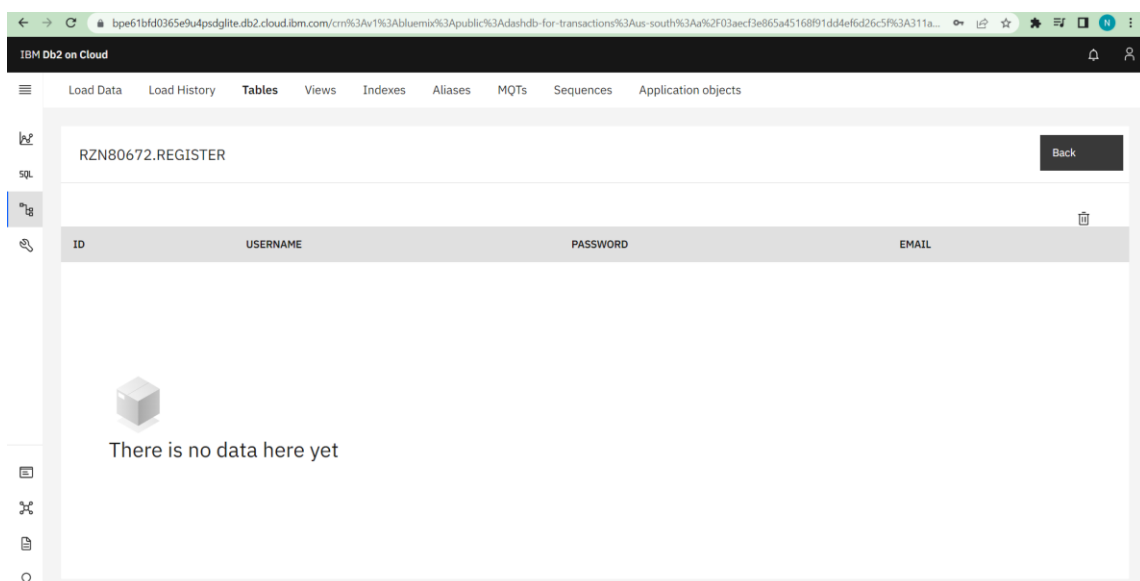


```
font-style: oblique;  
  
font-weight: bold;  
  
}
```

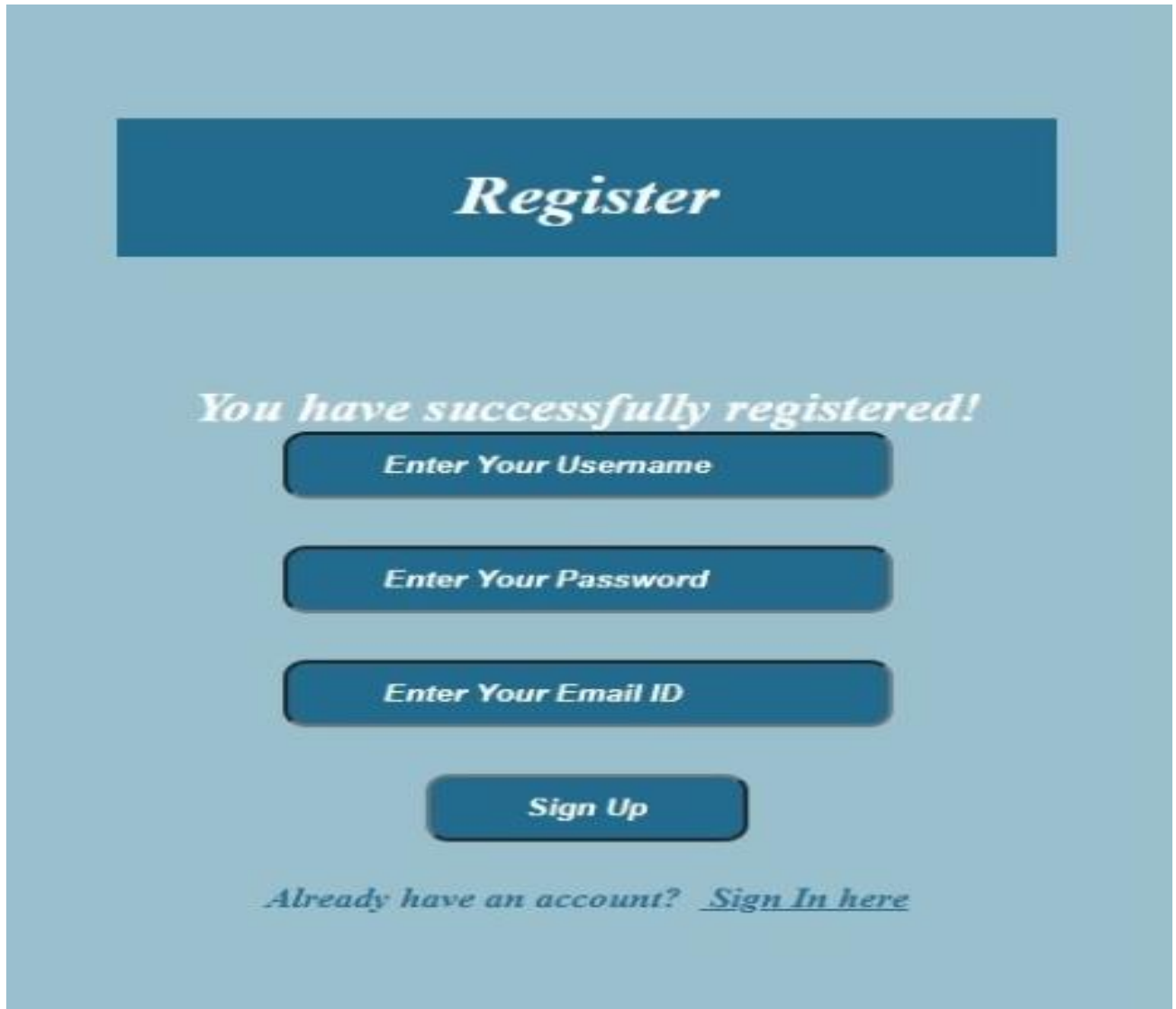
```
.word{  
  
color: #FFFFFF;  
  
font-style: oblique;  
  
font-weight: bold;  
  
}
```

```
.bottom{  
  
color: #236B8E;  
  
font-style: oblique;  
  
font-weight: bold;  
  
}
```

### Before registration:

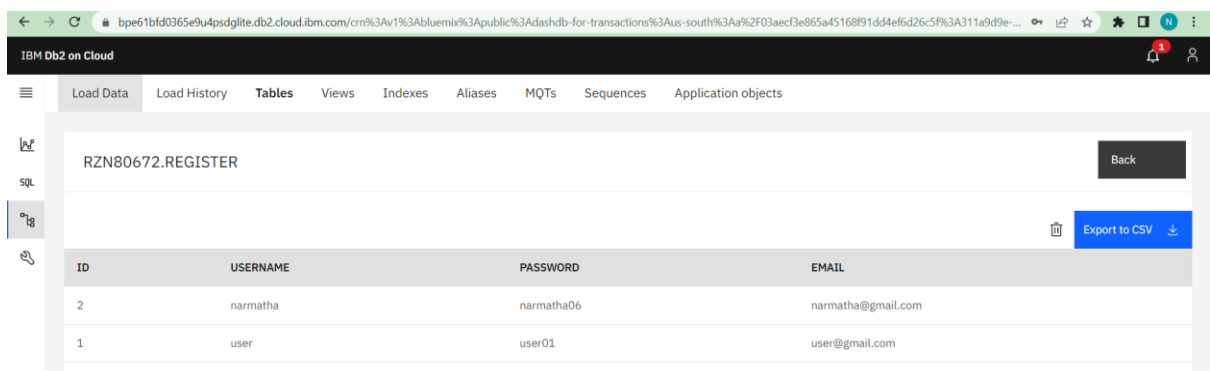


After registration:



The image shows a registration form on a light blue background. At the top, there is a dark blue button labeled "Register". Below it, a message states "You have successfully registered!". Underneath this message are three dark blue input fields labeled "Enter Your Username", "Enter Your Password", and "Enter Your Email ID". Below these fields is a dark blue button labeled "Sign Up". At the bottom, there is a link that says "Already have an account? [Sign In here](#)".

db2:



The image shows the IBM Db2 on Cloud interface. The top navigation bar includes "Load Data", "Load History", "Tables", "Views", "Indexes", "Aliases", "MQTs", "Sequences", and "Application objects". The "Tables" tab is selected, and the table "RZN80672.REGISTER" is displayed. The table has four columns: ID, USERNAME, PASSWORD, and EMAIL. The data is as follows:

ID	USERNAME	PASSWORD	EMAIL
2	narmatha	narmatha06	narmatha@gmail.com
1	user	user01	user@gmail.com

Buttons for "Back", "Export to CSV", and "Download" are visible in the top right corner of the table view.

Index page after successful logging:

