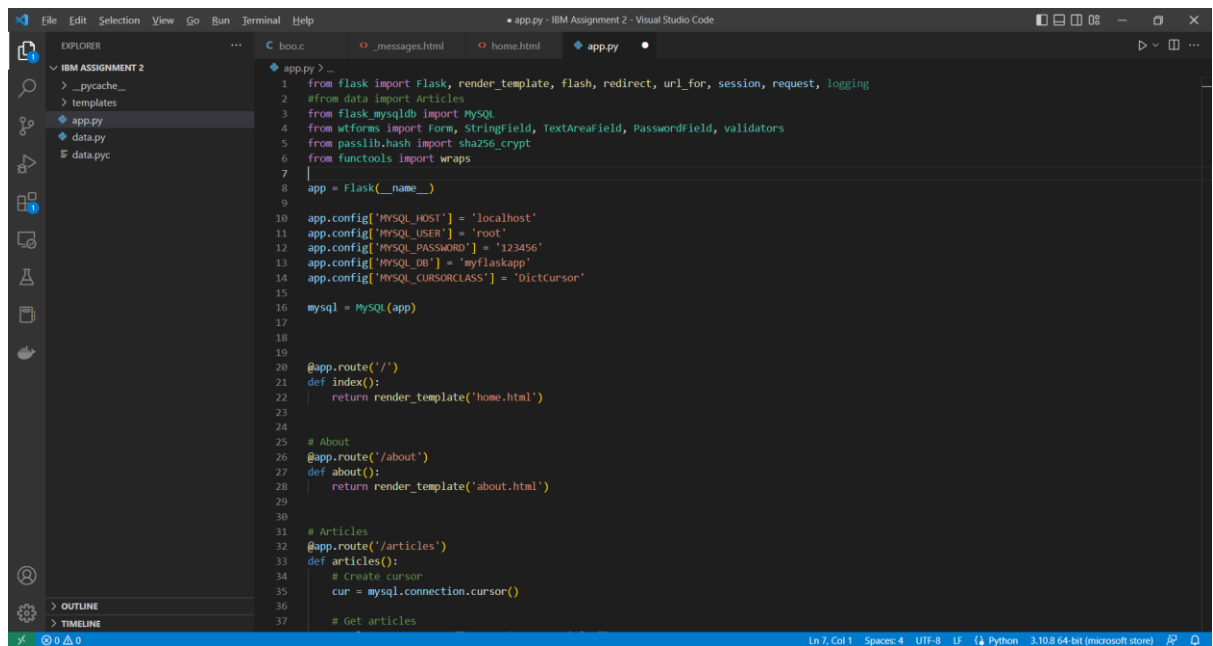


## Assignment -2

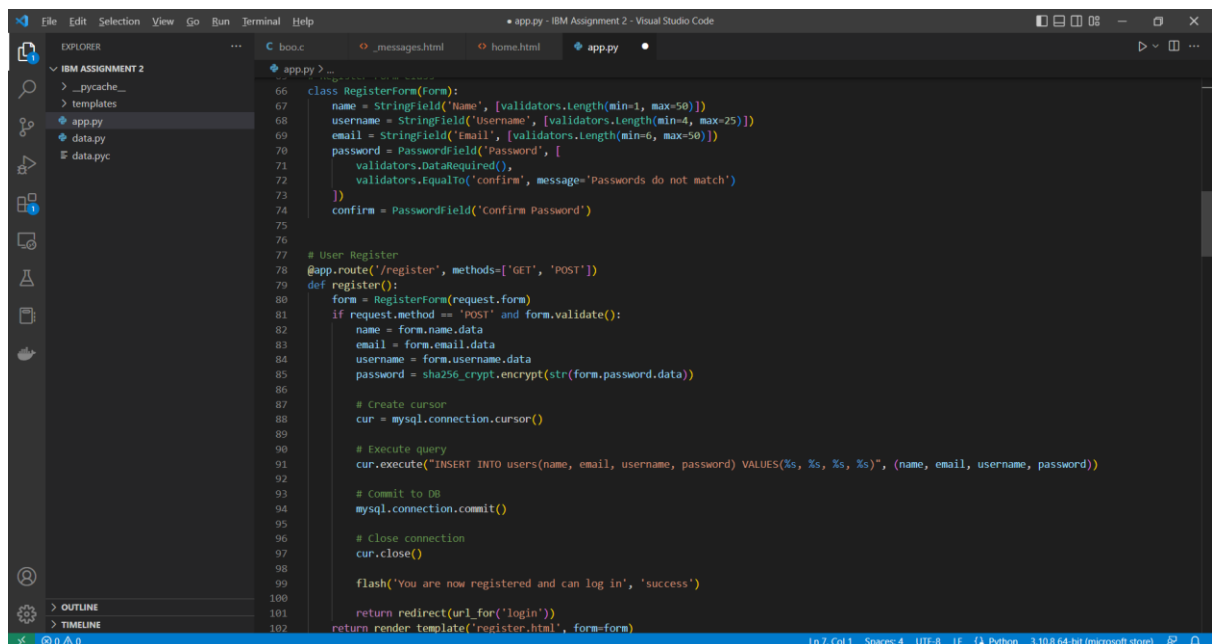
Assignment Date	25 September 2022
Student Name	Allan Jacob P
Team ID	PNT2022TMID00034
Maximum Marks	2 Marks

### 1) Create a Flask App



The screenshot shows the Visual Studio Code editor with a Python file named `app.py`. The code imports necessary modules for Flask, MySQL, and form validation. It configures the application with database credentials and sets up routes for the index, about, and articles pages. The `articles` route is partially implemented, showing a cursor creation and a query execution.

```
1 from flask import Flask, render_template, flash, redirect, url_for, session, request, logging
2 from data import Articles
3 from flask_mysql import MySQL
4 from wtforms import Form, StringField, TextAreaField, PasswordField, validators
5 from passlib.hash import sha256_crypt
6 from functools import wraps
7
8 app = Flask(__name__)
9
10 app.config['MYSQL_HOST'] = 'localhost'
11 app.config['MYSQL_USER'] = 'root'
12 app.config['MYSQL_PASSWORD'] = '123456'
13 app.config['MYSQL_DB'] = 'myflaskapp'
14 app.config['MYSQL_CURSORCLASS'] = 'DictCursor'
15
16 mysql = MySQL(app)
17
18
19
20 @app.route('/')
21 def index():
22     return render_template('home.html')
23
24
25 # About
26 @app.route('/about')
27 def about():
28     return render_template('about.html')
29
30
31 # Articles
32 @app.route('/articles')
33 def articles():
34     # Create cursor
35     cur = mysql.connection.cursor()
36
37     # Get articles
```

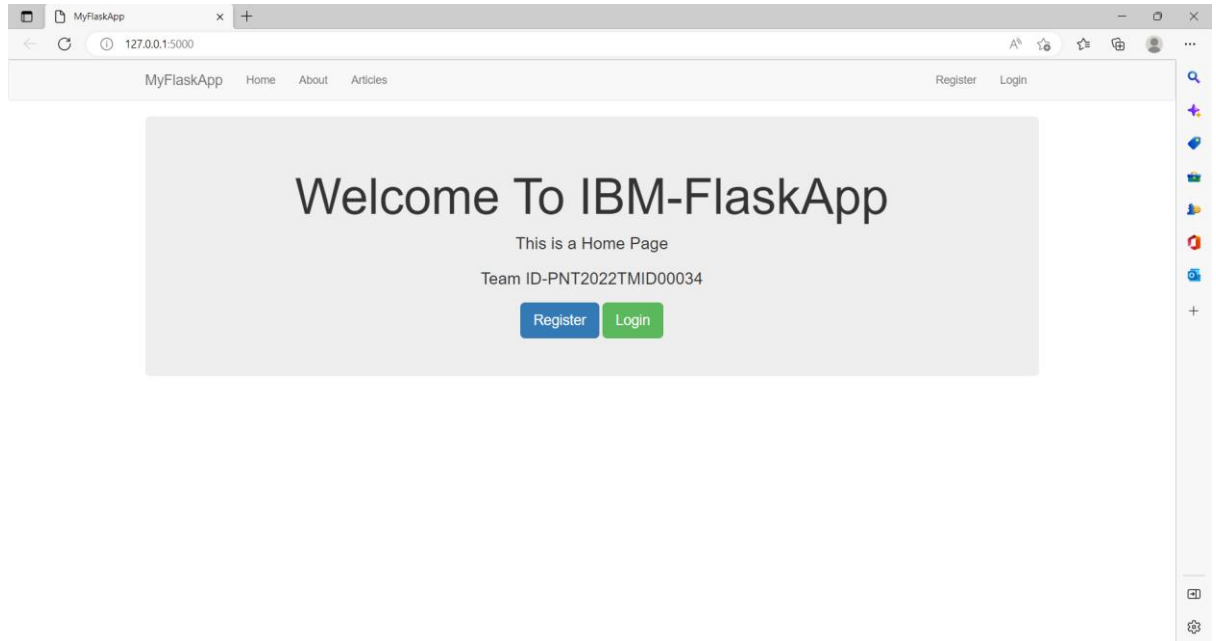


The screenshot shows the continuation of the `app.py` file. It defines a `RegisterForm` class with fields for name, username, email, and password, along with validation rules. The `register` route is implemented to handle POST requests, validate the form, hash the password, and insert the new user into the database using a cursor. It also includes a flash message and a redirect to the login page upon successful registration.

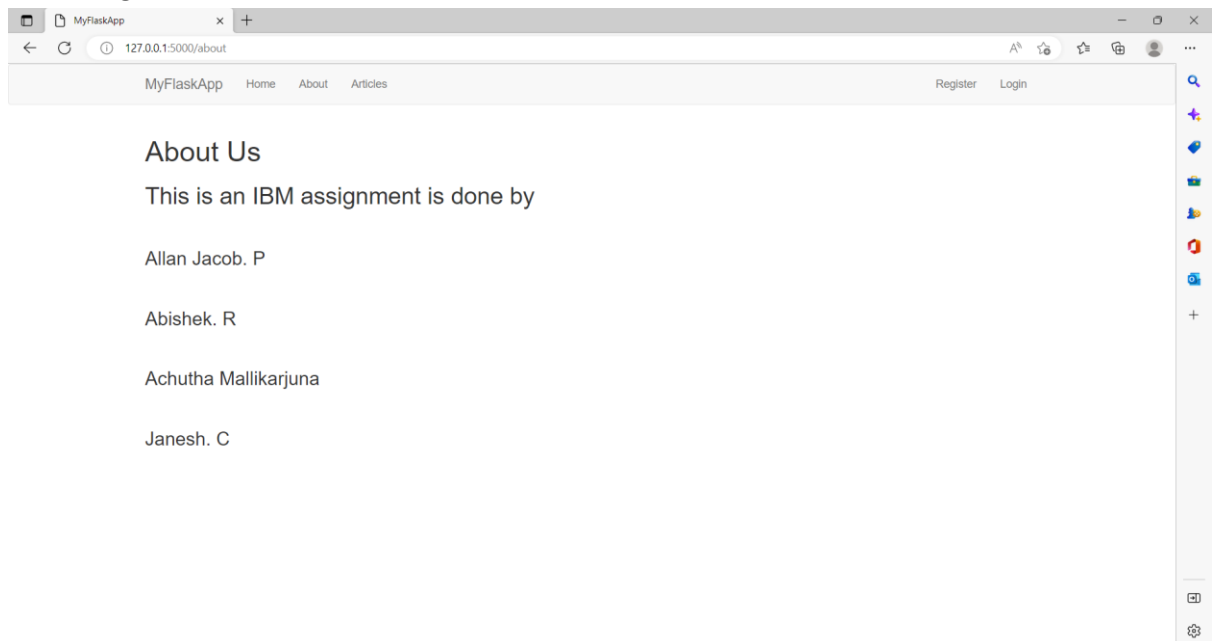
```
66 class RegisterForm(form):
67     name = StringField('Name', [validators.Length(min=1, max=50)])
68     username = StringField('Username', [validators.Length(min=4, max=25)])
69     email = StringField('Email', [validators.Length(min=6, max=50)])
70     password = PasswordField('Password', [
71         validators.DataRequired(),
72         validators.EqualTo('confirm', message='Passwords do not match')
73     ])
74     confirm = PasswordField('Confirm Password')
75
76
77 # User Register
78 @app.route('/register', methods=['GET', 'POST'])
79 def register():
80     form = RegisterForm(request.form)
81     if request.method == 'POST' and form.validate():
82         name = form.name.data
83         email = form.email.data
84         username = form.username.data
85         password = sha256_crypt.encrypt(str(form.password.data))
86
87         # Create cursor
88         cur = mysql.connection.cursor()
89
90         # Execute query
91         cur.execute("INSERT INTO users(name, email, username, password) VALUES(%s, %s, %s, %s)", (name, email, username, password))
92
93         # Commit to DB
94         mysql.connection.commit()
95
96         # Close connection
97         cur.close()
98
99         flash('You are now registered and can log in', 'success')
100
101         return redirect(url_for('login'))
102     return render_template('register.html', form=form)
```

## 2) Add Home Page and About Page

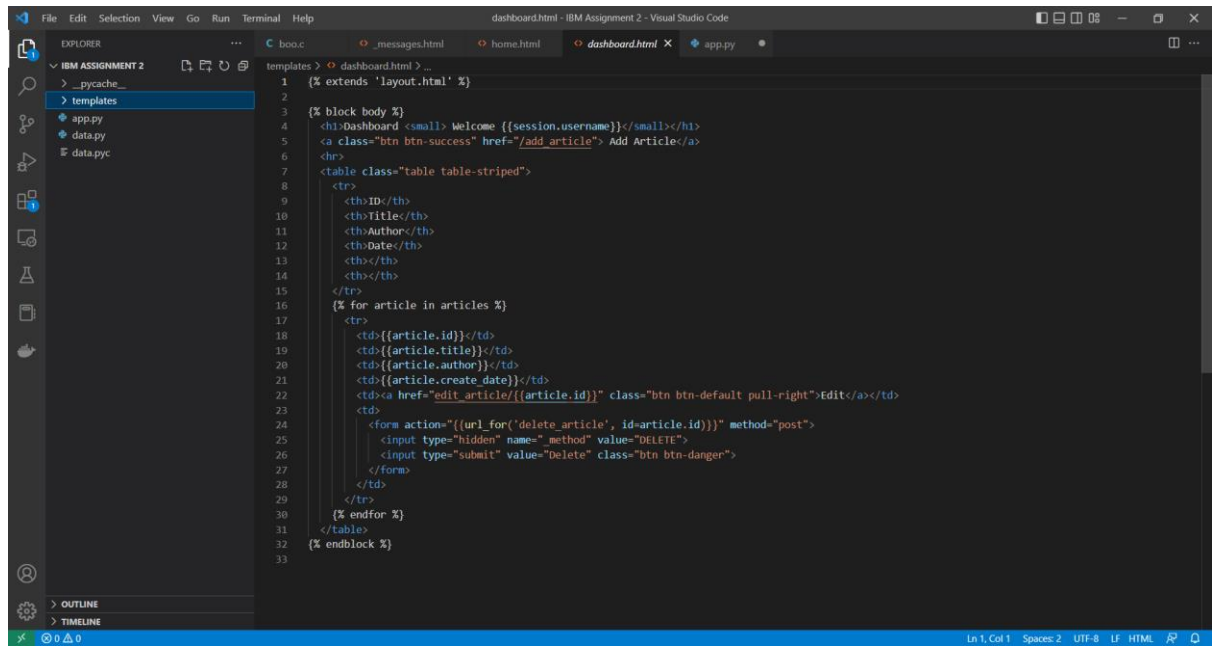
### Home Page



### About Page

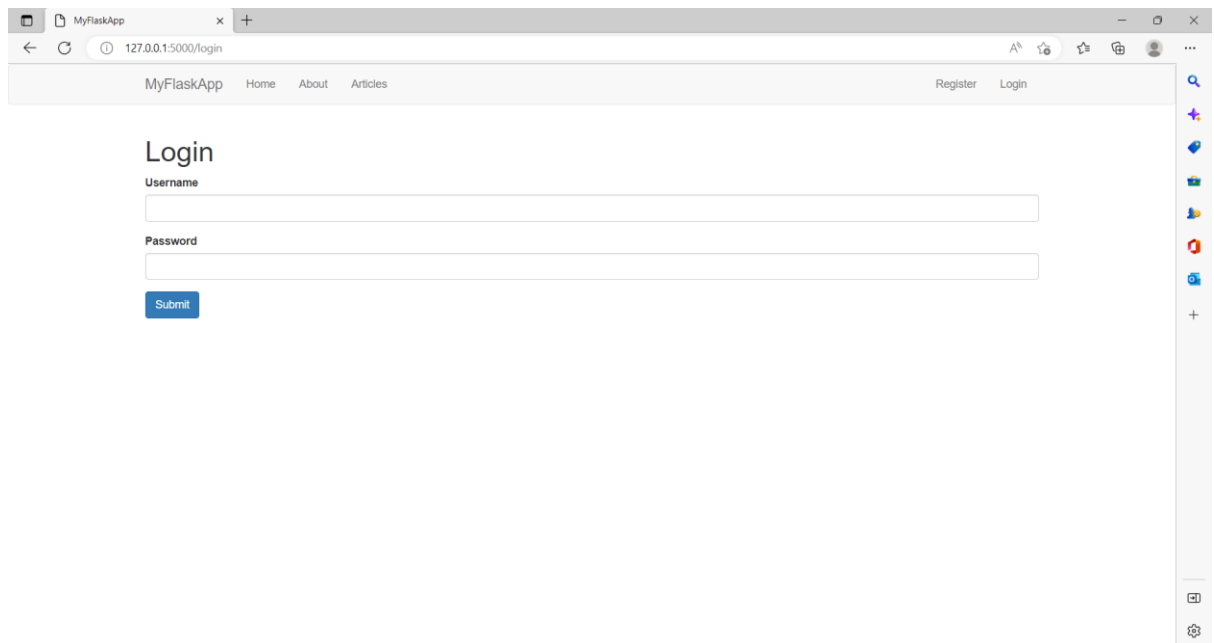


### 3) Add a Bootstrap



The screenshot shows the Visual Studio Code editor with the file `dashboard.html` open. The file is a Jinja2 template for a dashboard. It includes a header section with a welcome message and a button to add a new article. Below this is a table with columns for ID, Title, Author, and Date. The table contains a loop of articles, each with a row containing the article's ID, title, author, and date. Each row also includes an 'Edit' button and a 'Delete' button. The 'Delete' button is a form with a hidden method of 'DELETE' and a submit button labeled 'Delete'.

```
1 {% extends 'layout.html' %}
2
3 {% block body %}
4 <h1>Dashboard <small> Welcome {{session.username}}</small></h1>
5 <a class="btn btn-success" href="/add_article"> Add Article</a>
6 <hr>
7 <table class="table table-striped">
8 <tr>
9 <th>ID</th>
10 <th>Title</th>
11 <th>Author</th>
12 <th>Date</th>
13 <th></th>
14 </tr>
15 <tr>
16 {% for article in articles %}
17 <tr>
18 <td>{{article.id}}</td>
19 <td>{{article.title}}</td>
20 <td>{{article.author}}</td>
21 <td>{{article.create_date}}</td>
22 <td><a href="/edit_article/{{article.id}}" class="btn btn-default pull-right">Edit</a></td>
23 <td>
24 <form action="{{url_for('delete_article', id=article.id)}}" method="post">
25 <input type="hidden" name="_method" value="DELETE">
26 <input type="submit" value="Delete" class="btn btn-danger">
27 </form>
28 </td>
29 </tr>
30 {% endfor %}
31 </table>
32 {% endblock %}
33
```



#### 4) Add sign-in page and database connectivity

The screenshot shows a web browser window with the address bar displaying '127.0.0.1:5000/register'. The page title is 'MyFlaskApp' and the navigation bar includes links for 'Home', 'About', 'Articles', 'Register', and 'Login'. The main content area is titled 'Register' and contains a form with five input fields: 'Name', 'Email', 'Username', 'Password', and 'Confirm Password'. A blue 'Submit' button is located at the bottom of the form.

The screenshot shows the Visual Studio Code editor with the 'app.py' file open. The code implements a login route and a login function. The login route is defined as `@app.route('/login', methods=['GET', 'POST'])`. The login function checks if the request method is 'POST', retrieves form fields for 'username' and 'password', creates a cursor, and executes a SQL query to select the user by username. It then compares the stored password with the candidate password using `sha256_crypt.verify`. If the passwords match, it sets session variables for 'logged\_in' and 'username', flashes a success message, and redirects to the dashboard. Otherwise, it returns an error message and renders the login.html template.

```
105 # User login
106 @app.route('/login', methods=['GET', 'POST'])
107 def login():
108     if request.method == 'POST':
109         # Get Form Fields
110         username = request.form['username']
111         password_candidate = request.form['password']
112
113         # Create cursor
114         cur = mysql.connection.cursor()
115
116         # Get user by username
117         result = cur.execute("SELECT * FROM users WHERE username = %s", (username))
118
119         if result > 0:
120             # Get stored hash
121             data = cur.fetchone()
122             password = data['password']
123
124             # Compare Passwords
125             if sha256_crypt.verify(password_candidate, password):
126                 # Passed
127                 session['logged_in'] = True
128                 session['username'] = username
129
130                 flash('You are now logged in', 'success')
131                 return redirect(url_for('dashboard'))
132             else:
133                 error = 'Invalid login'
134                 return render_template('login.html', error=error)
135             # Close connection
136             cur.close()
137         else:
138             error = 'Username not found'
139             return render_template('login.html', error=error)
140
141 return render_template('login.html')
```