

## SPRINT 4

### PROJECT DELIVERABLES

#### (Flask Code and Deployment)

**Implementing the web page for collecting the data from the user and deploy the model to make predictions for the user inputs**

Team ID	PNT2022TMID52665
Project Name	Efficient Water Quality Analysis and Prediction Using Machine Learning

#### App.py: (Flask File)

Developed with Visual Studio Code

```
home.html X login.html app.py X register.html dashboard.html
app.py > ...
1 from flask import Flask, render_template, url_for, redirect, request
2 from flask_sqlalchemy import SQLAlchemy
3 from flask_login import UserMixin, login_user, LoginManager, login_required, logout_user, current_user
4 from flask_wtf import FlaskForm
5 from wtforms import StringField, PasswordField, SubmitField, IntegerField, FloatField
6 from wtforms.validators import InputRequired, Length, ValidationError
7 from flask_bcrypt import Bcrypt
8 import pickle
9 import numpy as np
10 import sklearn
11
12 model = pickle.load(open("wqi.pkl", "rb"))
13
14 app = Flask(__name__)
15 db = SQLAlchemy(app)
16 bcrypt = Bcrypt(app)
17 app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///database.db'
18 app.config['SECRET_KEY'] = 'thisisasecretkey'
19
20
21 login_manager = LoginManager()
22 login_manager.init_app(app)
23 login_manager.login_view = 'login'
24
25
26 @login_manager.user_loader
27 def load_user(user_id):
28     return User.query.get(int(user_id))
29
30
31 class User(db.Model, UserMixin):
32     id = db.Column(db.Integer, primary_key=True)
33     username = db.Column(db.String(20), nullable=False, unique=True)
34     password = db.Column(db.String(80), nullable=False)
35
```

```
home.html login.html app.py X register.html dashboard.html
app.py > ...

36 class WaterQualityIndex(db.Model):
37     id = db.Column(db.Integer, primary_key=True)
38     StationCode = db.Column(db.Integer, nullable=False)
39     State = db.Column(db.String(40), nullable=False)
40     Temp = db.Column(db.Float, nullable=False)
41     do = db.Column(db.Float, nullable=False)
42     ph = db.Column(db.Float, nullable=False)
43     co = db.Column(db.Integer, nullable=False)
44     bod = db.Column(db.Float, nullable=False)
45     na = db.Column(db.Float, nullable=False)
46     tc = db.Column(db.Integer, nullable=False)
47     Year = db.Column(db.Integer, nullable=False)
48     WQI = db.Column(db.Float, nullable=False)
49
50 class RegisterForm(FlaskForm):
51     username = StringField(validators=[
52         InputRequired(), Length(min=4, max=20)], render_kw={"placeholder": "Username"})
53
54     password = PasswordField(validators=[
55         InputRequired(), Length(min=8, max=20)], render_kw={"placeholder": "Password"})
56
57     submit = SubmitField('Register')
58
59     def validate_username(self, username):
60         existing_user_username = User.query.filter_by(
61             username=username.data).first()
62         if existing_user_username:
63             raise ValidationError(
64                 'That username already exists. Please choose a different one.')
65
66 class WQIForm(FlaskForm):
67     StationCode = IntegerField(validators=[
68         InputRequired()], render_kw={"placeholder": "Station Code"})
69
70     State = StringField(validators=[
71         InputRequired(), Length(min=8, max=20)], render_kw={"placeholder": "State"})
72     Temp = FloatField(validators=[
73         InputRequired()], render_kw={"placeholder": "Temp"})
74     do = FloatField(validators=[
75         InputRequired()], render_kw={"placeholder": "D.O"})
76     ph = FloatField(validators=[
77         InputRequired()], render_kw={"placeholder": "PH"})
78     co = IntegerField(validators=[
79         InputRequired()], render_kw={"placeholder": "Conductivity"})
80     bod = FloatField(validators=[
81         InputRequired()], render_kw={"placeholder": "B.O.D"})
82     na = FloatField(validators=[
83         InputRequired()], render_kw={"placeholder": "Nitratenen"})
84     tc = IntegerField(validators=[
85         InputRequired()], render_kw={"placeholder": "Coliform"})
86     Year = IntegerField(validators=[
87         InputRequired()], render_kw={"placeholder": "Year"})
88     submit = SubmitField('Predict')
89
90 class LoginForm(FlaskForm):
91     username = StringField(validators=[
92         InputRequired(), Length(min=4, max=20)], render_kw={"placeholder": "Username"})
93
94     password = PasswordField(validators=[
95         InputRequired(), Length(min=8, max=20)], render_kw={"placeholder": "Password"})
96
97     submit = SubmitField('Login')
98
99
100 @app.route('/')
101 def home():
102     return render_template('home.html')
```

```

106 @app.route('/login', methods=['GET', 'POST'])
107 def login():
108     form = LoginForm()
109     if form.validate_on_submit():
110         user = User.query.filter_by(username=form.username.data).first()
111         if user:
112             if bcrypt.check_password_hash(user.password, form.password.data):
113                 login_user(user)
114                 return redirect(url_for('dashboard'))
115     return render_template('login.html', form=form)
116
117
118 @app.route('/dashboard', methods=['GET', 'POST'])
119 @login_required
120 def dashboard():
121     form = WQIForm()
122     if form.validate_on_submit():
123         feature_val = []
124         feature_val.append(form.Temp.data)
125         feature_val.append(form.do.data)
126         feature_val.append(form.ph.data)
127         feature_val.append(form.co.data)
128         feature_val.append(form.bod.data)
129         feature_val.append(form.na.data)
130         feature_val.append(form.tc.data)
131         float_features = [float(x) for x in feature_val]
132         features = [np.array(float_features)]
133         prediction = model.predict(features)
134         new_data = WaterQualityIndex(StationCode=form.StationCode.data, State=form.State.data, Temp=form.Temp.data)
135         db.session.add(new_data)
136         db.session.commit()
137         return render_template('dashboard.html', form=form, prediction_text = "The water quality index is {}".format(prediction))
138     return render_template('dashboard.html', form=form)
139
140
141 @app.route('/logout', methods=['GET', 'POST'])
142 @login_required
143 def logout():
144     logout_user()
145     return redirect(url_for('login'))
146
147
148 @app.route('/register', methods=['GET', 'POST'])
149 def register():
150     form = RegisterForm()
151     if form.validate_on_submit():
152         hashed_password = bcrypt.generate_password_hash(form.password.data)
153         new_user = User(username=form.username.data, password=hashed_password)
154         db.session.add(new_user)
155         db.session.commit()
156         return redirect(url_for('login'))
157     return render_template('register.html', form=form)
158
159
160 if __name__ == '__main__':
161     app.run(debug=True)
162
163

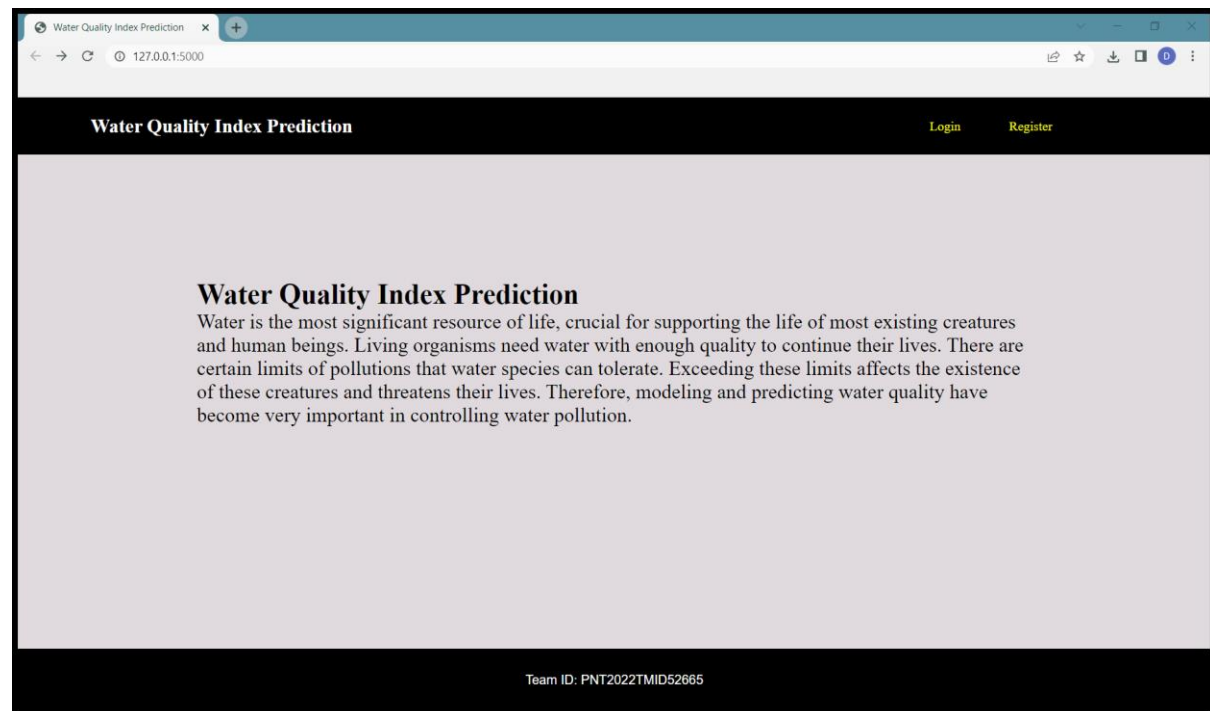
```

## Terminal:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  JUPYTER

* Serving Flask app 'app' (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Running on http://127.0.0.1:5000 (Press CTRL+C to quit)
* Restarting with stat
C:\Users\HP\Desktop\IBM_Project\venv\lib\site-packages\sklearn\base.py:329: UserWarning: Trying to unpickle estimator DecisionTreeRegressor from version 0.23.2 when using version 1.1.3. This might lead to breaking code or invalid results. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
warnings.warn(
C:\Users\HP\Desktop\IBM_Project\venv\lib\site-packages\sklearn\base.py:329: UserWarning: Trying to unpickle estimator RandomForestRegressor from version 0.23.2 when using version 1.1.3. This might lead to breaking code or invalid results. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
warnings.warn(
C:\Users\HP\Desktop\IBM_Project\venv\lib\site-packages\flask_sqlalchemy\_init_.py:851: UserWarning: Neither SQLALCHEMY_DATABASE_URI nor SQLALCHEMY_BINDS is set. Defaulting SQLALCHEMY_DATABASE_URI to "sqlite:///memory:".
warnings.warn(
C:\Users\HP\Desktop\IBM_Project\venv\lib\site-packages\flask_sqlalchemy\_init_.py:872: FSADeprecationWarning: SQLAlchemy_TRACK_MODIFICATIONS adds significant overhead and will be disabled by default in the future. Set it to True or False to suppress this warning.
warnings.warn(FSADeprecationWarning(
* Debugger is active!
* Debugger PIN: 119-411-840
```

## Home Page:



## Login Page:

The screenshot shows a web browser window with the title 'User Login' and the URL '127.0.0.1:5000/login'. The page has a black header with the text 'Water Quality Index Prediction'. The main content area has a light purple background and contains a 'Login' form. The form has two input fields: 'Username' and 'Password', followed by a green 'Login' button. Below the button is a red link that says 'Don't have an account? Sign Up'. The footer is black and contains the text 'Team ID: PNT2022TMID52665'.

Water Quality Index Prediction

### Login

Username

Password

Login

[Don't have an account? Sign Up](#)

Team ID: PNT2022TMID52665

## Register Page:

The screenshot shows a web browser window with the title 'User Registration' and the URL '127.0.0.1:5000/register'. The page has a black header with the text 'Water Quality Index Prediction'. The main content area has a light purple background and contains a 'Sign Up' form. The form has two input fields: 'Username' and 'Password', followed by a green 'Register' button. Below the button is a red link that says 'Already have an account? Log In'. The footer is black and contains the text 'Team ID: PNT2022TMID52665'.

Water Quality Index Prediction

### Sign Up

Username

Password

Register

[Already have an account? Log In](#)

Team ID: PNT2022TMID52665

## Dashboard Page:

A screenshot of a web browser displaying a dashboard titled "Water Quality Index Prediction". The browser's address bar shows "127.0.0.1:5000/dashboard". The dashboard has a black header with the title on the left and a "Logout" link on the right. The main content area has a light purple background and contains the text "Enter the parameters to predict the Water Quality Index". Below this text is a form with ten input fields arranged in two columns: "Station Code", "State", "Temp", "D.O", "PH", "Conductivity", "B.O.D", "Nitratenen", "Coliform", and "Year". A green "Predict" button is centered below the input fields. At the bottom of the dashboard, a black footer displays "Team ID: PNT2022TMID52665".

Water Quality Index Prediction [Logout](#)

Enter the parameters to predict the Water Quality Index

Station Code	State
Temp	D.O
PH	Conductivity
B.O.D	Nitratenen
Coliform	Year

Predict

Team ID: PNT2022TMID52665

A screenshot of the same "Water Quality Index Prediction" dashboard, but with sample data entered into the input fields. The browser address bar and dashboard header/footer are identical to the first screenshot. The input fields now contain the following values: "Station Code" is 1011, "State" is Maharashtra, "Temp" is 22.4, "D.O" is 9.8, "PH" is 7.0, "Conductivity" is 245, "B.O.D" is 0.5, "Nitratenen" is 0.0, "Coliform" is 10, and "Year" is 2021. The green "Predict" button remains at the bottom of the form.

Water Quality Index Prediction [Logout](#)

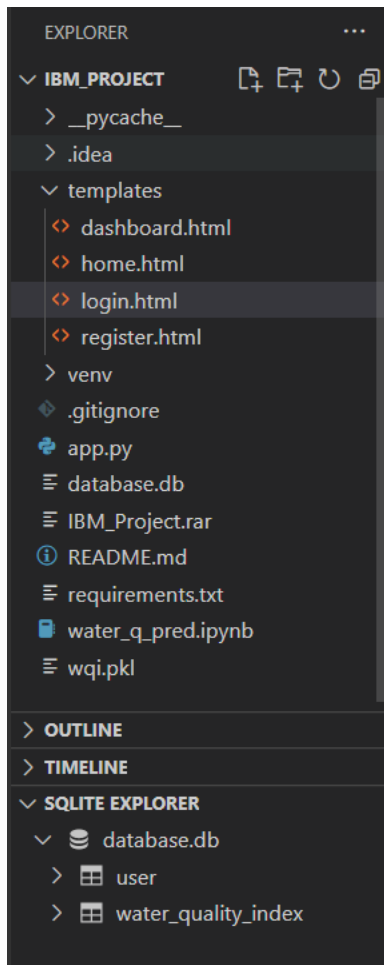
Enter the parameters to predict the Water Quality Index

1011	Maharashtra
22.4	9.8
7.0	245
0.5	0.0
10	2021

Predict

Team ID: PNT2022TMID52665

## Folder Structure:



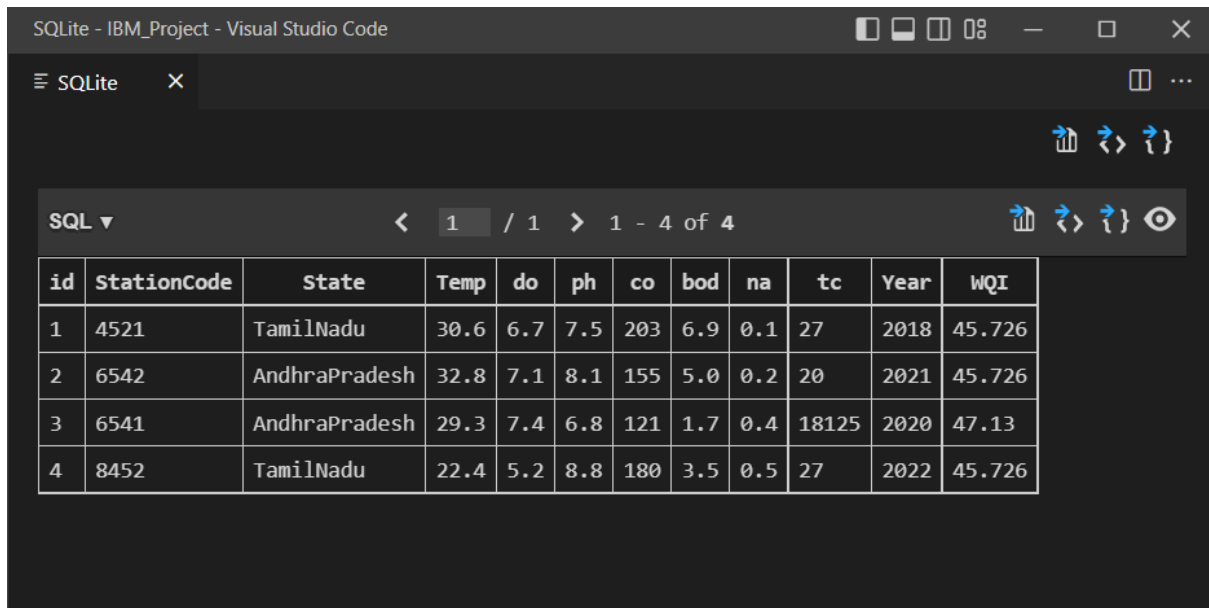
## Database:

### user Table

The SQLite Explorer shows the following data for the user table:

id	username	password
1	divya	\$2b\$12\$4gontoUSl4/Cr1B3sSiy2ezKCpq0gVIt.8YoIsMoRz1WtYd2BtfYm
2	user1	\$2b\$12\$4bitAN/YTTrqLP55.SBK6.jqazi4J.5UcremUDrgxD06M6UwWPuke

## water\_quality\_index Table:

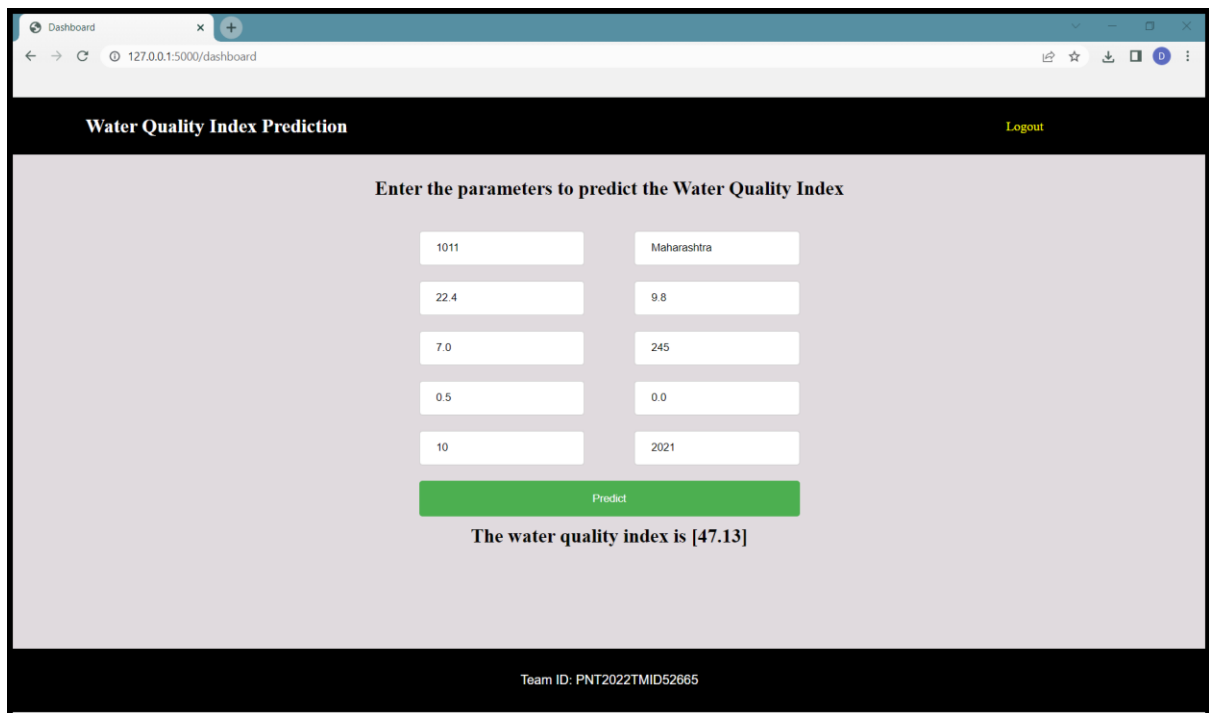


The screenshot shows a SQLite database viewer interface. At the top, it says 'SQLite - IBM\_Project - Visual Studio Code'. Below that, there's a tab labeled 'SQLite'. The main area displays a table with 12 columns: id, StationCode, State, Temp, do, ph, co, bod, na, tc, Year, and WQI. There are 4 rows of data. The table is displayed in a grid format with a dark background and white text.

id	StationCode	State	Temp	do	ph	co	bod	na	tc	Year	WQI
1	4521	TamilNadu	30.6	6.7	7.5	203	6.9	0.1	27	2018	45.726
2	6542	AndhraPradesh	32.8	7.1	8.1	155	5.0	0.2	20	2021	45.726
3	6541	AndhraPradesh	29.3	7.4	6.8	121	1.7	0.4	18125	2020	47.13
4	8452	TamilNadu	22.4	5.2	8.8	180	3.5	0.5	27	2022	45.726

## Test Case:

Marginal: (WQI Value 45-64) – Water quality is frequently impaired; conditions often depart from desirable levels.



The screenshot shows a web application titled 'Water Quality Index Prediction'. It has a 'Logout' link in the top right corner. The main content area has a heading 'Enter the parameters to predict the Water Quality Index'. Below this, there are two columns of input fields. The left column contains five numeric input fields with values: 1011, 22.4, 7.0, 0.5, and 10. The right column contains five input fields, the first of which is a dropdown menu showing 'Maharashtra', followed by four numeric input fields with values: 9.8, 245, 0.0, and 2021. Below these input fields is a green 'Predict' button. Under the button, it says 'The water quality index is [47.13]'. At the bottom of the page, it says 'Team ID: PNT2022TMID52665'.

Water Quality Index Prediction [Logout](#)

Enter the parameters to predict the Water Quality Index

1011	Maharashtra
22.4	9.8
7.0	245
0.5	0.0
10	2021

[Predict](#)

The water quality index is [47.13]

Team ID: PNT2022TMID52665