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 | login.html | app.py | register.html | dashboard.html

app.py > ...

```python
from flask import Flask, render_template, url_for, redirect, request
from flask_sqlalchemy import SQLAlchemy
from flask_login import UserMixin, login_user, LoginManager, login_required, logout_user, current_user
from flask_wtf import FlaskForm
from wtforms import StringField, PasswordField, SubmitField, IntegerField, FloatField
from wtforms.validators import InputRequired, Length, ValidationError
from flask_bcrypt import Bcrypt
import pickle
import numpy as np
import sklearn

model = pickle.load(open("wqi.pkl", "rb"))

app = Flask(__name__)
db = SQLAlchemy(app)
bcrypt = Bcrypt(app)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///database.db'
app.config['SECRET_KEY'] = 'thisisasecretkey'


login_manager = LoginManager()
login_manager.init_app(app)
login_manager.login_view = 'login'


@login_manager.user_loader
def load_user(user_id):
    return User.query.get(int(user_id))


class User(db.Model, UserMixin):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(20), nullable=False, unique=True)
    password = db.Column(db.String(80), nullable=False)

```

home.html | login.html | app.py | register.html | dashboard.html

app.py > ...

```python
class WaterQualityIndex(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    StationCode = db.Column(db.Integer, nullable=False)
    State = db.Column(db.String(40), nullable=False)
    Temp = db.Column(db.Float, nullable=False)
    do = db.Column(db.Float, nullable=False)
    ph = db.Column(db.Float, nullable=False)
    co = db.Column(db.Integer, nullable=False)
    bod = db.Column(db.Float, nullable=False)
    na = db.Column(db.Float, nullable=False)
    tc = db.Column(db.Integer, nullable=False)
    Year = db.Column(db.Integer, nullable=False)
    WQI = db.Column(db.Float, nullable=False)

class RegisterForm(FlaskForm):
    username = StringField(validators=[
                        InputRequired(), Length(min=4, max=20)], render_kw={"placeholder": "Username"})

    password = PasswordField(validators=[
                         InputRequired(), Length(min=8, max=20)], render_kw={"placeholder": "Password"})

    submit = SubmitField('Register')

    def validate_username(self, username):
        existing_user_username = User.query.filter_by(
            username=username.data).first()
        if existing_user_username:
            raise ValidationError(
                'That username already exists. Please choose a different one.')
```

```python
class WQIForm(FlaskForm):
    StationCode = IntegerField(validators=[
                        InputRequired()], render_kw={"placeholder": "Station Code"})

    State = StringField(validators=[
                         InputRequired(), Length(min=8, max=20)], render_kw={"placeholder": "State"})
    Temp = FloatField(validators=[
                         InputRequired()], render_kw={"placeholder": "Temp"})
    do = FloatField(validators=[
                         InputRequired()], render_kw={"placeholder": "D.O"})
    ph = FloatField(validators=[
                         InputRequired()], render_kw={"placeholder": "PH"})
    co = IntegerField(validators=[
                         InputRequired()], render_kw={"placeholder": "Conductivity"})
    bod = FloatField(validators=[
                         InputRequired()], render_kw={"placeholder": "B.O.D"})
    na = FloatField(validators=[
                         InputRequired()], render_kw={"placeholder": "Nitratenen"})
    tc = IntegerField(validators=[
                         InputRequired()], render_kw={"placeholder": "Coliform"})
    Year = IntegerField(validators=[
                         InputRequired()], render_kw={"placeholder": "Year"})
    submit = SubmitField('Predict')

class LoginForm(FlaskForm):
    username = StringField(validators=[
                        InputRequired(), Length(min=4, max=20)], render_kw={"placeholder": "Username"})

    password = PasswordField(validators=[
                         InputRequired(), Length(min=8, max=20)], render_kw={"placeholder": "Password"})

    submit = SubmitField('Login')


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

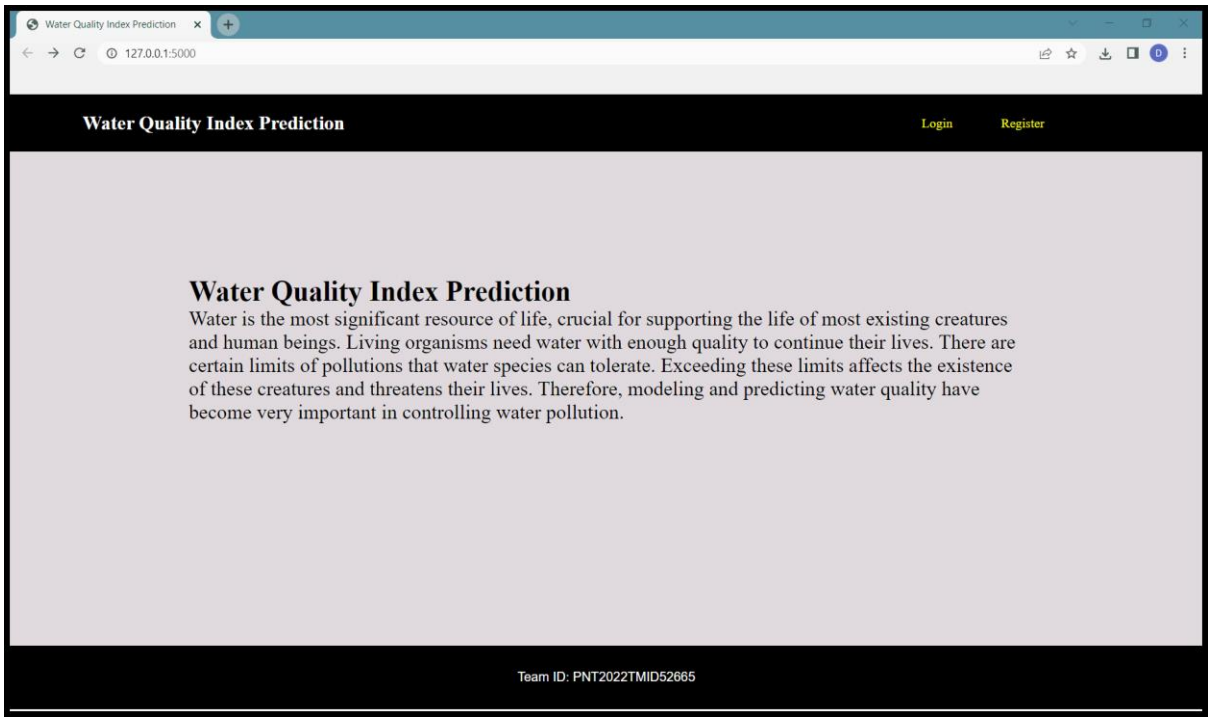
```python
@app.route('/login', methods=['GET', 'POST'])
def login():
    form = LoginForm()
    if form.validate_on_submit():
        user = User.query.filter_by(username=form.username.data).first()
        if user:
            if bcrypt.check_password_hash(user.password, form.password.data):
                login_user(user)
                return redirect(url_for('dashboard'))
    return render_template('login.html', form=form)


@app.route('/dashboard', methods=['GET', 'POST'])
@login_required
def dashboard():
    form = WQIForm()
    if form.validate_on_submit():
        feature_val = []
        feature_val.append(form.Temp.data)
        feature_val.append(form.do.data)
        feature_val.append(form.ph.data)
        feature_val.append(form.co.data)
        feature_val.append(form.bod.data)
        feature_val.append(form.na.data)
        feature_val.append(form.tc.data)
        float_features = [float(x) for x in feature_val]
        features = [np.array(float_features)]
        prediction = model.predict(features)
        new_data = WaterQualityIndex(StationCode=form.StationCode.data, State =form.State.data, Temp=form.Temp
        db.session.add(new_data)
        db.session.commit()
        return render_template('dashboard.html',form=form, prediction_text = "The water quality index is {}".fo

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

```

```python
@app.route('/logout', methods=['GET', 'POST'])
@login_required
def logout():
    logout_user()
    return redirect(url_for('login'))


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

    if form.validate_on_submit():
        hashed_password = bcrypt.generate_password_hash(form.password.data)
        new_user = User(username=form.username.data, password=hashed_password)
        db.session.add(new_user)
        db.session.commit()
        return redirect(url_for('login'))

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

if __name__ == "__main__":
    app.run(debug=True)

```

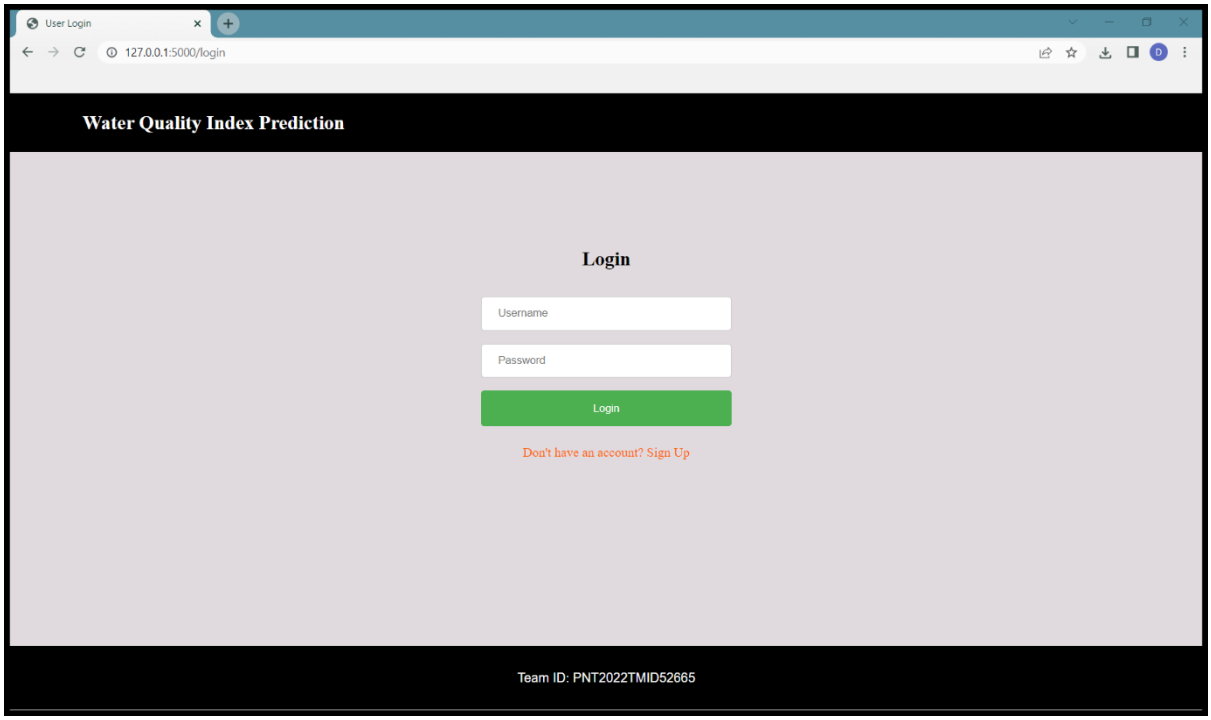
Terminal:

```
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 De
cisionTreeRegressor from version 0.23.2 when using version 1.1.3. This might lead to breaking code or invalid results. U
se 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 Ra
ndomForestRegressor from version 0.23.2 when using version 1.1.3. This might lead to breaking code or invalid results. U
se 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: SQLALCHE
MY_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
```

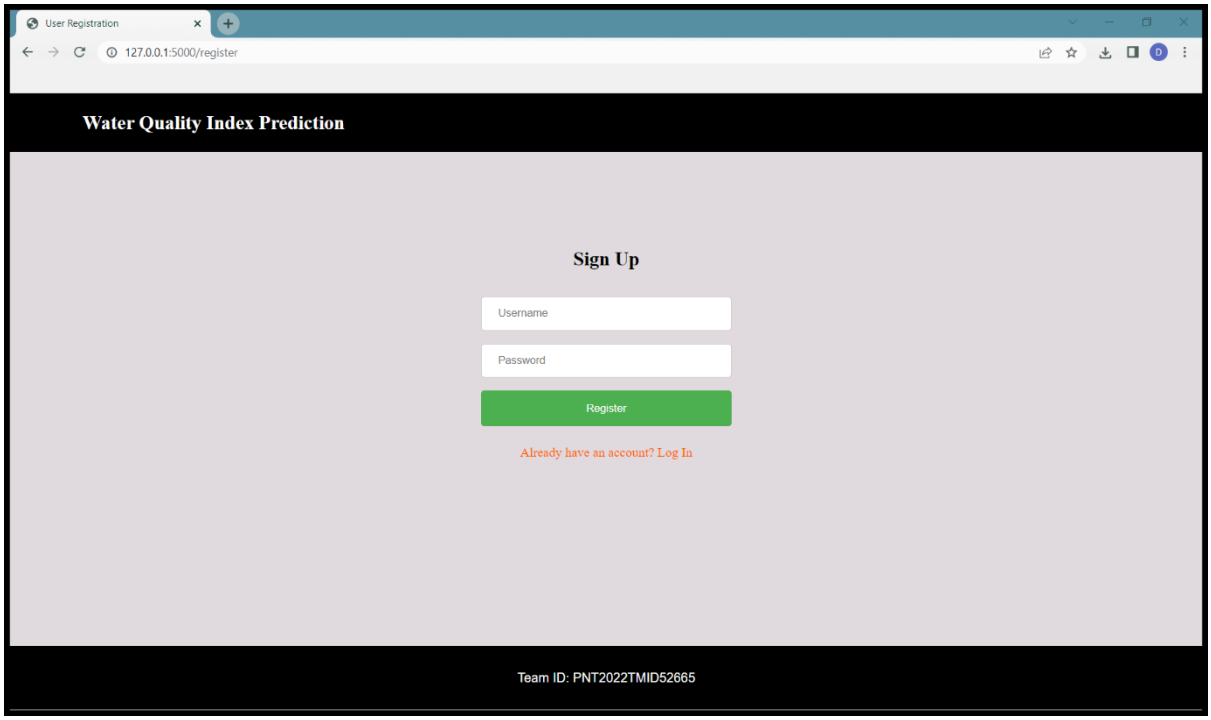
Home Page:



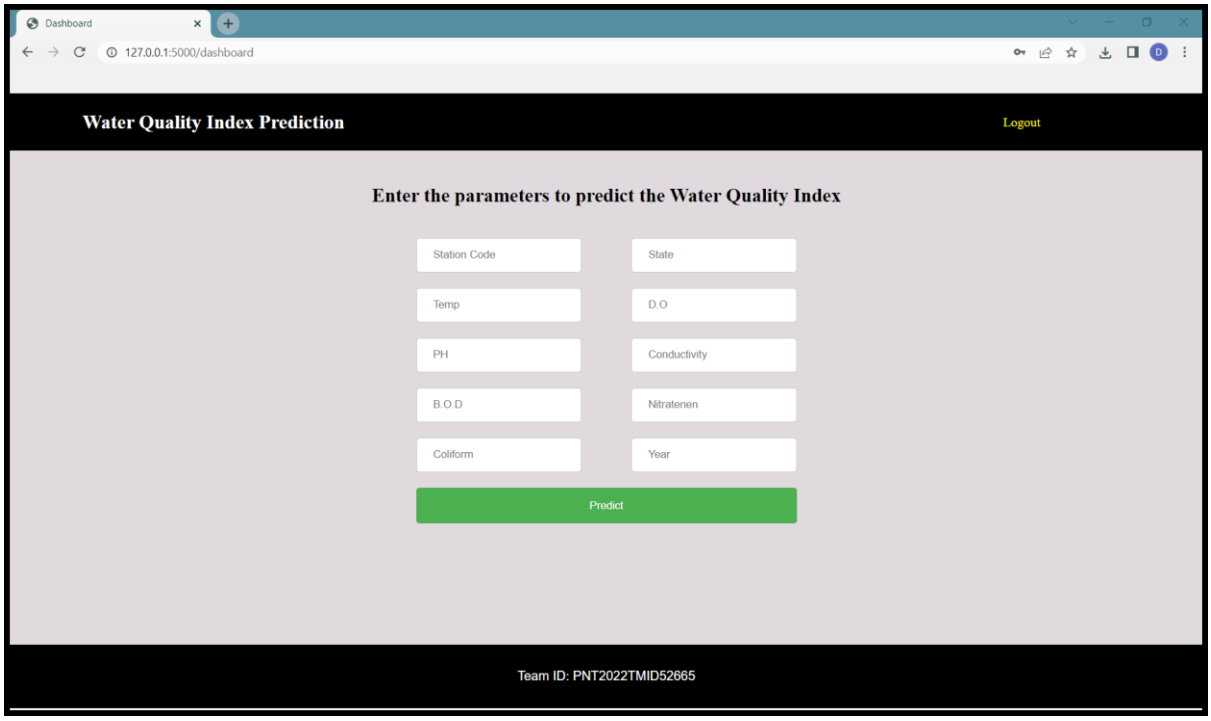
Login Page:

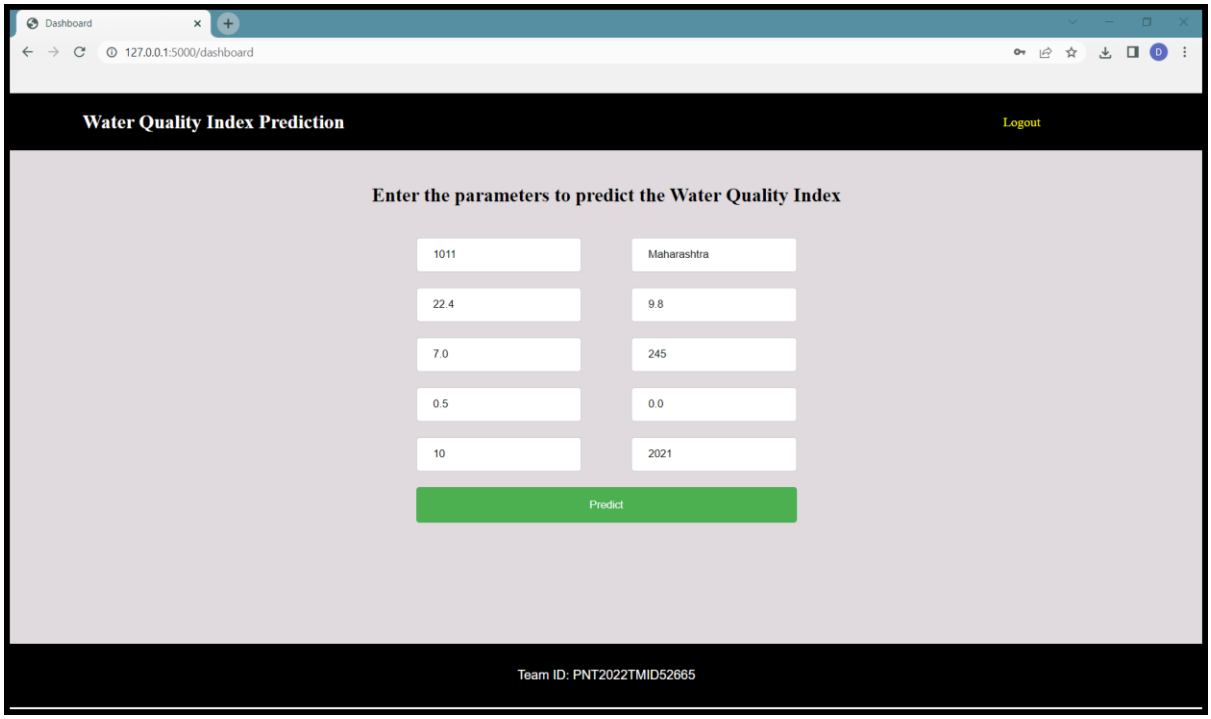


Register Page:

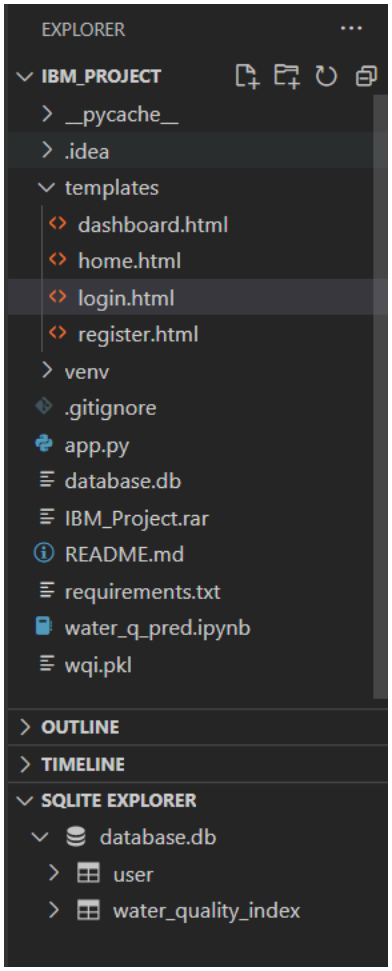


Dashboard Page:



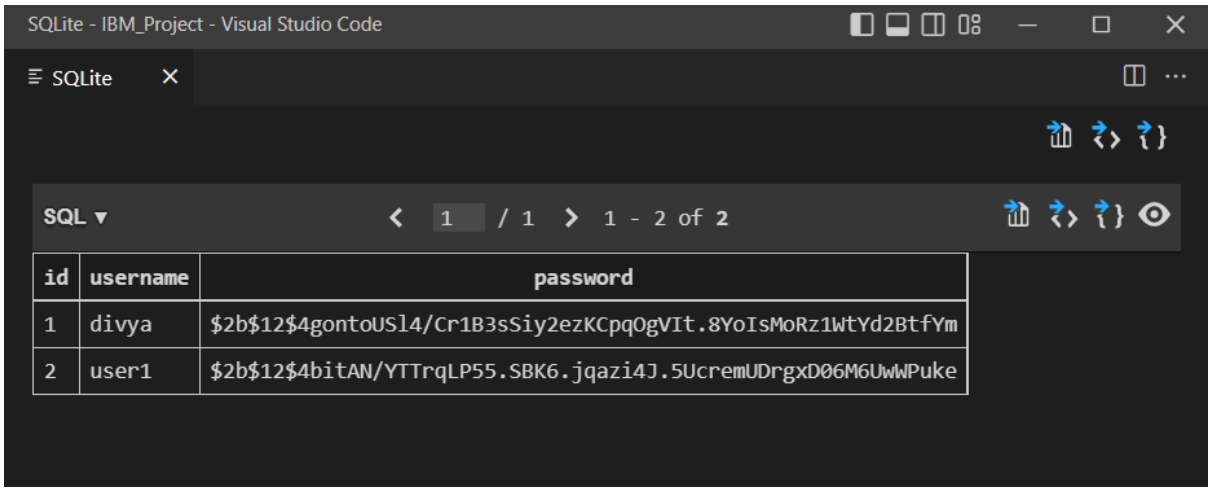


Folder Structure:



Database:

user Table



id	username	password
1	divya	$2b$12$4gontoUSl4/Cr1B3sSiy2ezKCpqOgVIt.8YoIsMoRz1WtYd2BtfYm
2	user1	$2b$12$4bitAN/YTTrqLP55.SBK6.jqazi4J.5UcremUDrgxD06M6UwWPuke

water_quality_index Table:

SQLite - IBM_Project - Visual Studio Code

SQLite

SQL ▾ 1 / 1 1 - 4 of 4

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.