<table>
<tr><th>Task</th><th>Answer</th><th>Mark</th></tr>
<tr><td>3.1</td><td>

Mark Breakdown:

1. Username and password fields
2. Screen for viewing of students’ recorded marks:
    a. Exactly 4 input fields with labels present (award mark even if not displayed in a table format)
        i. Index number of student
        ii. Student’s class
        iii. Date of presentation
        iv. Marks obtained
3. Screen for input of student’s marks
    a. Exactly 4 input fields with labels present
        i. Student Index No. :
        ii. Class:
        iii. Date of presentation:
        iv. Marks obtained:
4. Tabular form of presenting results
5. Buttons for user to login, create new record, and view
6. Text and user navigation does not confuse user and/or contradict task requirements
7. UI doesn’t prevent user from navigating through all 3 main features of the app

Suggested answer:

login.html

```
<!doctype html>
<html>
<head><title>Teacher's app</title></head>
<body>
  Please login to use the app.
<br><br>
    <form action="/" method="POST">
        Username<input type="text" name="username1"><br>
        Password<input type="text" name="password1"><br>
        <br><input type="submit">
    </form>
</body>
</html>
```

view.html

```
<!DOCTYPE html>
<html>
<head><title>Teacher's app home page</title>

  <style>
  table {
```

</td><td>[7]</td></tr>
</table>

```
    font-family: arial, sans-serif;
    border-collapse: collapse;
    width: 100%;
  }

  td, th {
    border: 1px solid #dddddd;
    text-align: left;
    padding: 8px;
  }

  tr:nth-child(even) {
    background-color: #dddddd;
  }
  </style>

</head>
<body>

<h1>
View Records</h1>

    <table>
      <tr>
        <th>Index number of student</th>
        <th>Student's class</th>
        <th>Date of presentation</th>
        <th>Marks obtained</th>
      </tr>

        {% for line in rows %}
<tr>
  {% for item in line %}
            <td> {{ item }}  </td>
{% endfor %}
</tr>
{% endfor %}

    </table>
<br><br>

<h1>
Create Record</h1>
<br>
Please enter student and presentation details.
<br><br>

<form action="/submit" method="POST">
    Student Index No.: <input type="text"
name="student_id"><br>
    Class: <input type="text" name="student_class"><br>
    Date of presentation<input type="text"
name="prsentation_date"><br>
    Marks Obtained: <input type="text"
name="student_marks"><br>
<br><br>
```

```
    <input type="submit">
</form>

</body>
</html>
```

3.2

Breakdown of marks

1. Creation of 3 tables
2. Composite key of StudentIndex, Class, PresentationDate in Presentation table
3. Foreign key of Class in other tables
4. Foreign key of TeacherUsername in other tables
5. Correct columns in Presentations table
6. Correct columns TeacherAccounts table
7. Correct columns in ClassAllocation table
8. Correct data type setup for all table columns

Suggested answer:

```
CREATE TABLE TeacherAccounts (
     TeacherUsername TEXT(20) NOT NULL PRIMARY KEY,
     TeacherPassword TEXT(20),
     FOREIGN KEY (TeacherUsername)
          REFERENCES ClassAllocation(TeacherUsername)

);

CREATE TABLE Presentations (
     StudentIndex INTEGER(2) NOT NULL,
     Class TEXT(6),
     PresentationDate INTEGER(8),
     Marks REAL,
     PRIMARY KEY (StudentIndex, Class, PresentationDate),
     FOREIGN KEY (Class)
          REFERENCES ClassAllocation(Class)
);


CREATE TABLE ClassAllocation (
     Class TEXT(6) NOT NULL PRIMARY KEY,
     TeacherUsername TEXT(20),
     FOREIGN KEY (TeacherUsername)
          REFERENCES TeacherAccounts(TeacherUsername)
);
```

[8]

3.3

Breakdown of marks

1. Correct use of SQL insertion syntax
2. Correct values used to insert into the tables
3. Correct data type used (no inverted commas around integer values)

4. No other SQL syntax errors

[4]

Suggested answer:

```
INSERT INTO TeacherAccounts  (TeacherUsername,
TeacherPassword) VALUES ('mr_raj', 'cr53aYJP');
INSERT INTO TeacherAccounts  (TeacherUsername,
TeacherPassword) VALUES ('mr_james', '8orjqiZc');
INSERT INTO TeacherAccounts  (TeacherUsername,
TeacherPassword) VALUES ('mdm_rahayu', '7iqndCjW');

INSERT INTO ClassAllocation  (Class, TeacherUsername)
VALUES ('19S306', 'mr_raj');
INSERT INTO ClassAllocation  (Class, TeacherUsername)
VALUES ('19S301', 'mr_james');
INSERT INTO ClassAllocation  (Class, TeacherUsername)
VALUES ('19S302', 'mdm_rahayu');
INSERT INTO ClassAllocation  (Class, TeacherUsername)
VALUES ('19S304', 'mr_james');

INSERT INTO Presentations  (StudentIndex, Class,
PresentationDate, Marks) VALUES (3, '19S306', 20200315, 95);
INSERT INTO Presentations  (StudentIndex, Class,
PresentationDate, Marks) VALUES (24, '19S301', 20200315,
60);
INSERT INTO Presentations  (StudentIndex, Class,
PresentationDate, Marks) VALUES (2, '19S302', 20200315,
35.5);
INSERT INTO Presentations  (StudentIndex, Class,
PresentationDate, Marks) VALUES (11, '19S304', 20200325,
60);
```

3.4

Breakdown of marks

1. Flask application begins running without errors
2. Correct use of the flask.Flask.route() decorator to associate functions with either a static path or a path containing one or more variable sections (i.e. `@app.route(...)`)
3. Correct use of `render_template` function or `return` to display html
4. Creation of SQLite connection to DB
5. Commits DB connection (where necessary)
6. Close DB connection
7. SQL query for teacher's login is correct
8. SQL query for viewing of marks features the correct syntax and columns
9. App ensures that user cannot see other teachers' records of classes they don't teach (can be done via SQL or python)
10. SQL query for creating new record is correct
11. UI of HTML+CSS form for login credentials displayed by app correctly

12. App processing of login credentials works successfully (only accepts valid credentials)
13. UI of HTML+CSS for student presentation records displayed by app correctly (eg: tabular form)
14. App processing to display student presentation records works successfully
15. UI of HTML+CSS form for creating new presentation record displayed by app correctly
16. App processing to insert new record works successfully

[16]

Note:

- No marks deducted for issues such as
    - Security vulnerabilities of login component
    - Lack of usability (Eg: ease of viewing the updated records after inserting new record)
    - Lack of user input validation

Suggested answer:

```
import flask
from flask import render_template, request
app = flask.Flask(__name__)
import sqlite3

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

#Exit because user needs to provide login credentials first
    if request.method == 'GET':
        return render_template('login.html')

#Exit if user's login credentials is incorrect
    db = sqlite3.connect('teacherapp.db')
    cur = db.execute('SELECT * FROM TeacherAccounts')
    counter = 0
    for row in cur:
        if row[0]==request.form['username1'] and
row[1]==request.form['password1']:
            counter=1
    if counter ==0:
        return "Login invalid"
    db.close()

#Only display records of classes taught by the teacher of
this account
    db = sqlite3.connect('teacherapp.db')
    cur = db.execute("SELECT * from Presentations LEFT JOIN
ClassAllocation ON Presentations.Class =
ClassAllocation.Class WHERE TeacherUsername='"+
request.form['username1'] +  "';")
    rows = []

    for row in cur:
        temp = []
        temp.append(row[0])
        temp.append(row[1])
        temp.append(row[2])
        temp.append(row[3])
        rows.append(temp)
    db.close()

    return render_template('view.html',rows=rows)


@app.route('/submit', methods=['POST'])
def submit():
# insert a new student presentation record
    db = sqlite3.connect('teacherapp.db')
    db.execute('INSERT INTO Presentations  (StudentIndex,
Class, PresentationDate, Marks) VALUES (' +
request.form['student_id'] +',
'+request.form['student_class']+', '+
request.form['prsentation_date']+',
'+request.form['student_marks']+');')
    db.commit()
    db.close()

    return "Records updated. <a href='/'>Click here</a> back
to return."


if __name__ == '__main__':
    app.run()
```

3.5

Breakdown of marks

1. Usage of SUM() function
2. Usage of GROUP BY statement
3. Correct table selected
4. Correct fields used for SUM() and GROUP BY
5. SQL query works successfully

[5]

Suggested answer:

```
SELECT Class, SUM(Marks) FROM Presentations GROUP by Class;
```