

# MONTCLAIR STATE UNIVERSITY

## **FINAL PROJECT REPORT**

### **LLC TAX AND PAYMENT TRACKING SYSTEM**

#### **Course**

CSIT555\_01FA24 – Database Systems

FALL 2024

#### **SUBMITTED TO**

Professor Xiaofeng Li

#### **SUBMITTED BY**

Aaryaman Singh Patel

## INDEX

TITLE	Pg.no
<i>Abstract</i>	1
INTRODUCTION	1
SYSTEM DESIGN	1
1. Frontend Design	1
2. Backend Design	1
3. Database Design	1
4. Workflow and Interactions	1
IMPLEMENTATION	2
1. Frontend Implementation	2
• HTML	2
• CSS	5
• JavaScript	7
2. Backend Implementation	8
• Flask Framework	8
• Data Handling	8
3. Database Implementation	9
• SQLite	9
4. Dynamic Features	9
SQL SCHEMA	9
CONTROLLERS AND ENDPOINTS	10
PYTHON FLASK CODE AND LIST OF APIs (Source Code)	10
UI and WORKFLOW OF THE WEB APPLICATION	14
1. Details Section	15
2. All Payments Section	15
3. Popup	16
4. Delete confirmation popup	17
5. Payment summary section	17
VIDEO AND GITHUB LINKS	18

## **ABSTRACT:**

This project implements a tax and payment tracking system designed for small businesses to manage their tax payments efficiently. The system has a user friendly web interface for adding, updating and viewing payment records dynamically filtered in the help of due dates. Using python flask for the back end and Sqlite for the database, the system ensures better record keeping and provides real time summaries for better financial tracking.

## **INTRODUCTION:**

The Tax and Payment Tracking System Is a web based application designed to help small companies like LLCs and corporations in managing and tracking their tax payments efficiently. The application is built using Python flask as a backend framework, which allows the users to perform CURD operations (Create, Read, Update, Delete) on the record stored in the SQLite database. This application dynamically calculates taxes and gives a summary of payments based on the selected due dates. The software used for front end HTML, CSS and JavaScript to deliver a better user interface. The application ensures Seamless Management of financial data with the help of user friendly design principles.

## **SYSTEM DESIGN:**

The architecture integrates the following key components.

### **1. Frontend Design:**

- The user interface is built using HTML, CSS and JavaScript
- The CSS is designed to ensure responsiveness and dropdowns for selecting the options.
- JavaScript handles interactivity, such as fetching summary data dynamically and managing CRUD operations.

### **2. Backend Design:**

- The backend is implemented using Python flask.
- The flask routes handles various operations like inserting, updating, deleting and fetching payment records.
- A custom jinja2 template filter format dates dynamically for better presentation.

### **3. Database Design:**

- The database is SQLite
- A table named payments is used to store tax data including fields for company name, amount, payment date, status, due date and tax rate.
- Sql queries are used for performing the CURD operations.

### **4. Workflow and Interactions:**

- Users interact with the system with the web interface and performing operations as required.

- Actions such as editing or deleting trigger asynchronous operations through JavaScript which enables smooth user experiences.

## IMPLEMENTATION:

### 1. Frontend Implementation:

- **HTML:**

The structure of the web pages, forms for adding or editing payments and tables for displaying the payment summaries is created using html.

```

app.py  <> index.html  app.js  style.css
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Tax and Payment Tracking System</title>
7      <link rel="stylesheet" href="/static/style.css">
8  </head>
9  <body>
10     <div class="container">
11         <h1>LLC Tax and Payment Tracking System</h1>
12         <form id="taxForm" action="/submit" method="POST">
13             <label for="company">Company:</label>
14             <input type="text" id="company" name="company" placeholder="Company name" required>
15             <label for="amount">Amount:</label>
16             <input type="number" id="amount" name="amount" step="0.01" placeholder="Enter amount" required>
17             <label for="paymentDate">Payment Date:</label>
18             <input type="date" id="paymentDate" name="paymentDate">
19             <label for="status">Status:</label>
20             <select id="status" name="status" required>
21                 <option value="paid">Paid</option>
22                 <option value="unpaid">Unpaid</option>
23             </select>
24             <label for="dueDate">Due Date:</label>
25             <select id="dueDate" name="dueDate" required>
26                 <option value="">Select the due dates</option>
27                 {% for due_date in due_dates %}
28                 <option value="{{ due_date }}">{{ due_date }}</option>
29                 {% endfor %}
30             </select>

```

```

31         <label for="taxRate">Tax Rate:</label>
32         <input type="number" id="taxRate" name="taxRate" step="0.01" placeholder="Enter tax rate" required>
33         <button type="submit">Save</button>
34     </form>
35
36     <h3>Payment Summary</h3>
37     <select id="summaryDueDate">
38         <option value="">Select the due dates</option>
39         {% for due_date in due_dates %}
40         <option value="{{ due_date }}">{{ due_date }}</option>
41         {% endfor %}
42     </select>
43     <div id="summary"></div>
44
45     <h3>All Payments</h3>
46     <table class="table table-striped">
47         <thead>
48             <tr>
49                 <th>ID</th>
50                 <th>Company</th>
51                 <th>Amount</th>
52                 <th>Payment Date</th>
53                 <th>Status</th>
54                 <th>Due Date</th>
55                 <th>Actions</th>
56             </tr>
57         </thead>

```

```

58         <tbody>
59             {% for records in records %}
60             <tr>
61                 <td>{{ records[0] }}</td>
62                 <td>{{ records[1] }}</td>
63                 <td>{{ records[2] }}</td>
64                 <td>{% if records[3] %}{{ records[3]|format_date }}{% else %}NA{% endif %}</td>
65                 <td>{{ records[4] }}</td>
66                 <td>{{ records[5] }}</td>
67                 <td>
68                     <button class="edit-btn" onclick="openEditPopup('{{ records[0] }}', '{{ records[1] }}', '{{ records[2] }}',
69                     {{ '{{ records[3] }}', '{{ records[4] }}', '{{ records[5] }}', '{{ records[6] }}' }}">Edit</button>
70                     <button class="delete-btn" onclick="deleteRecord('{{ records[0] }}')">Delete</button>
71                 </td>
72             </tr>
73             {% endfor %}
74         </tbody>
75     </table>
76 </div>

```

```

<div class="overlay" id="overlay"></div>
<div class="edit-popup" id="editPopup">
  <h2>Edit Details</h2>
  <form id="editForm">
    <input type="hidden" id="editId" name="editId">
    <label for="editCompany">Company:</label>
    <input type="text" id="editCompany" name="editCompany" required>
    <label for="editAmount">Amount:</label>
    <input type="number" id="editAmount" name="editAmount" step="0.01" required>
    <label for="editPaymentDate">Payment Date:</label>
    <input type="date" id="editPaymentDate" name="editPaymentDate" required>
    <label for="editStatus">Status:</label>
    <select id="editStatus" name="editStatus" required>
      <option value="paid">Paid</option>
      <option value="unpaid">Unpaid</option>
    </select>
    <label for="editDueDate">Due Date:</label>
    <select id="editDueDate" name="editDueDate" required>
      <option value="">Select Due Date</option>
      {% for due_date in due_dates %}
      <option value="{{ due_date }}">{{ due_date }}</option>
      {% endfor %}
    </select>
    <label for="editTaxRate">Tax Rate:</label>
    <input type="number" id="editTaxRate" name="editTaxRate" step="0.01" required>
    <button type="button" onclick="saveEdit()">Save</button>
    <button type="button" class="cancel-btn" onclick="closeEditPopup()">Cancel</button>
  </form>
</div>

```

```

<script src="/static/app.js"></script>
</body>
</html>

```

- **CSS:**

All the styling of the pages and design of the pages were done using the CSS styling.

```

1  body {
2      font-family: 'Roboto', Arial, sans-serif;
3      background-image: url('bg_tax.jpg');
4      background-size: cover;
5      background-repeat: no-repeat;
6      background-position: center;
7      margin: 0;
8      padding: 0;
9      color: #333;
10 }
11
12 .container {
13     max-width: 800px;
14     margin: 20px auto;
15     padding: 25px;
16     background-color: rgba(255, 255, 255, 0.9);
17     box-shadow: 0 4px 15px rgba(0, 0, 0, 0.2);
18     border-radius: 10px;
19 }
20
21 h1 {
22     text-align: center;
23     margin-bottom: 25px;
24     font-size: 2rem;
25     color: #28a745;
26 }
27
28 form {
29     max-width: 650px;
30     margin: 0 auto;

```

```

}

label {
    display: block;
    font-weight: bold;
    margin-bottom: 8px;
}

input[type="text"],
input[type="number"],
input[type="date"],
select,
button {
    width: 100%;
    padding: 10px;
    margin-bottom: 20px;
    border-radius: 8px;
    border: 1px solid #ddd;
    box-sizing: border-box;
    font-size: 1rem;
    transition: border-color 0.3s ease;
}

input:focus,
select:focus {
    border-color: #28a745;
    outline: none;
    box-shadow: 0 0 5px rgba(40, 167, 69, 0.3);
}

```

```

button {
    background-color: #28a745;
    color: #fff;
    border: none;
    padding: 12px;
    font-size: 1rem;
    border-radius: 8px;
    cursor: pointer;
    transition: background-color 0.3s ease;
}

button:hover {
    background-color: #218838;
}

button:active {
    transform: scale(0.98);
}

table {
    width: 100%;
    border-collapse: collapse;
    margin-top: 25px;
    font-size: 0.9rem;
}

th, td {
    border: 1px solid #ccc;
    padding: 12px;
    text-align: left;

```

```

th {
    background-color: #28a745;
    color: #fff;
    font-weight: bold;
}

tr:nth-child(even) {
    background-color: #f9f9f9;
}

.edit-btn {
    background-color: #28a745;
    color: #fff;
    padding: 8px 15px;
    border-radius: 5px;
    border: none;
    cursor: pointer;
    font-size: 0.9rem;
    transition: opacity 0.3s ease;
}

.delete-btn,
.cancel-btn {
    background-color: #dc3545;
    color: #fff;
    padding: 8px 15px;
    border-radius: 5px;
    border: none;
    cursor: pointer;
    font-size: 0.9rem;
}

```

```

    transition: opacity 0.3s ease;
  }

  .edit-btn:hover {
    opacity: 0.8;
  }

  .delete-btn:hover,
  .cancel-btn:hover {
    background-color: #a71d2a;
  }

  .overlay {
    display: none;
    position: fixed;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    background-color: rgba(0, 0, 0, 0.6);
    z-index: 999;
  }

  .edit-popup {
    display: none;
    position: fixed;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
    background-color: #fff;

```

```

    padding: 30px;
    border-radius: 10px;
    box-shadow: 0 6px 20px rgba(0, 0, 0, 0.2);
    z-index: 1000;
    width: 90%;
    max-width: 500px;
  }

```



- **JavaScript:**

JavaScript handles client-side interactivity. It enables

- i. Displaying payment summaries dynamically based on user selected due dates.
- ii. Popup modals for editing records
- iii. Asynchronous updates and deletion vis API calls.

```
async function fetchSummary() {
    const dueDateDropdown = document.getElementById('summaryDueDate');
    const dueDate = dueDateDropdown.value;

    const summaryDiv = document.getElementById('summary');
    if (!dueDate) {
        summaryDiv.innerHTML = '';
        return;
    }

    const response = await fetch(`/summary?dueDate=${dueDate}`);
    const data = await response.json();
    summaryDiv.innerHTML = data.html;
}

document.getElementById('summaryDueDate').addEventListener('change', fetchSummary);

function openEditPopup(id, company, amount, paymentDate, status, dueDate, taxRate) {
    document.getElementById('editId').value = id;
    document.getElementById('editCompany').value = company;
    document.getElementById('editAmount').value = amount;
    document.getElementById('editPaymentDate').value = paymentDate;
    document.getElementById('editStatus').value = status;
    document.getElementById('editDueDate').value = dueDate;
    document.getElementById('editTaxRate').value = taxRate;
    document.getElementById('editPopup').style.display = 'block';
    document.getElementById('overlay').style.display = 'block';
}
```

```

function closeEditPopup() {
    document.getElementById('editPopup').style.display = 'none';
    document.getElementById('overlay').style.display = 'none';
}

async function saveEdit() {
    const formData = new FormData(document.getElementById('editForm'));
    await fetch('/update', {
        method: 'POST',
        body: formData
    });
    location.reload(true);
    await fetchSummary();
    closeEditPopup();
}

async function deleteRecord(id) {
    if (confirm('Are you sure you want to delete this record?')) {
        await fetch(`/delete?id=${id}`, {
            method: 'DELETE'
        });
        location.reload(true);
        await fetchSummary();
    }
}

```

## 2. Backend Implementation:

- **Flask Framework:**

The backend of the system is built using Flask with the routes defined for each operation. Below are the routes used.

- `@app.route('/submit')`: This route handles the insertion of new payment records.
- `@app.route('/update')`: This route updates existing records.
- `@app.route('/delete')`: This route deletes records based on the user's action.
- `@app.route('/summary')`: Fetches filtered data for the summary table.

- **Data Handling:**

Flask receives all the users inputs from forms and processes them before interacting with the database.

### 3. Database Implementation:

- **SQLite:**

- i. The database is used for storing payment details. The schema includes fields like id, company, amount, payment\_date, status, due\_date and tax\_rate.
- ii. CRUD operations are performed with the help of SQL queries which are written in the Flask routes.

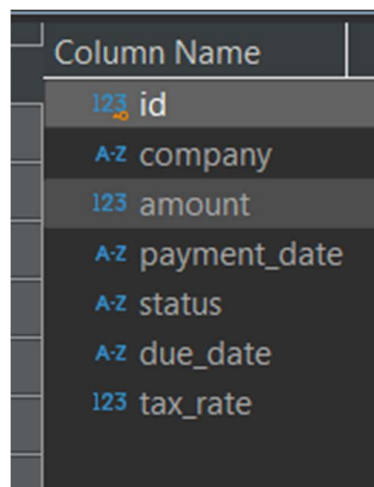
### 4. Dynamic Features:

- The due dates dropdown dynamically generates information based on that current year.
- The summary table won't appear until a specific due date is selected from the dropdown under the Payment Summary section.

## SQL SCHEMA

The database contains a table named payments Having the following schema:

- id : A primary key to uniquely identify each record
- company : Stores the company name
- amount : Stores payment amounts as a decimal value.
- payment\_date : Track the payment date in yyyy-mm-dd format.
- status : Indicates if a payment is paid or unpaid.
- due\_date : Tracks the tax payment due date.
- tax\_rate : Holds the tax rate as a decimal.



Column Name
id
company
amount
payment_date
status
due_date
tax_rate

**Payments table schema (Dbeaver)**

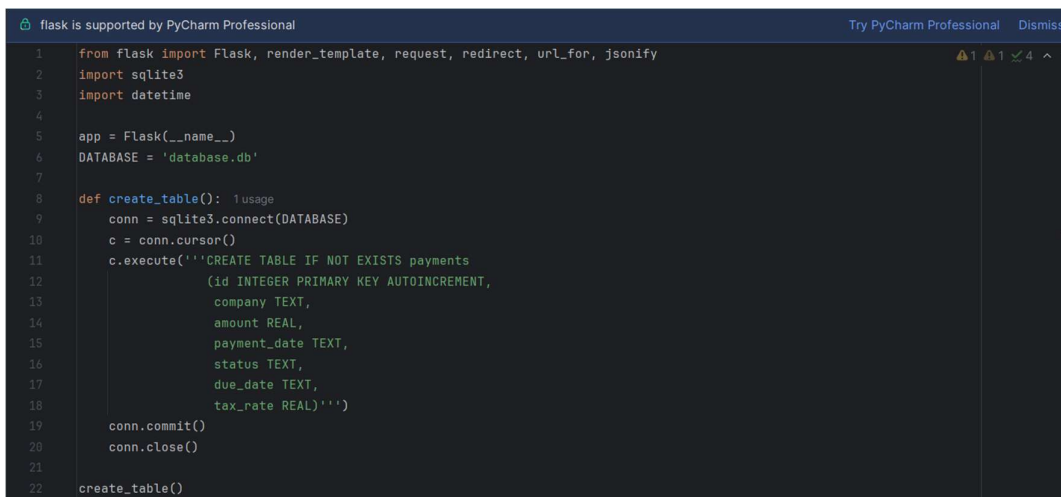
## CONTROLLERS AND ENDPOINTS

Controllers in flask handle routing and manage the applications logic.

- index : The main page, fetching and displaying all payment records.
- submit : Processes new payment entries via Post request.
- update : Updates an existing payments details.
- delete : Delete the payment record by identifying the id.
- summary : Fetches and philtres payment records by a selected due date

Endpoints are linked to their respective functions, ensuring smooth communication between the frontend and the backend.

## PYTHON FLASK CODE AND LIST OF APIs (SOURCE CODE)

A screenshot of the PyCharm Professional IDE interface. The top status bar indicates 'flask is supported by PyCharm Professional' and 'Try PyCharm Professional Dismiss'. The main editor area displays Python code for a Flask application. The code includes imports for Flask, render\_template, request, redirect, url\_for, jsonify, sqlite3, and datetime. It initializes a Flask app, sets a database path, and defines a create\_table function that connects to a SQLite database, creates a cursor, and executes a SQL statement to create a 'payments' table with columns for id, company, amount, payment\_date, status, due\_date, and tax\_rate. The function commits the transaction and closes the connection. The code is numbered from 1 to 22.

```
1 from flask import Flask, render_template, request, redirect, url_for, jsonify
2 import sqlite3
3 import datetime
4
5 app = Flask(__name__)
6 DATABASE = 'database.db'
7
8 def create_table():
9     conn = sqlite3.connect(DATABASE)
10    c = conn.cursor()
11    c.execute('CREATE TABLE IF NOT EXISTS payments
12              (id INTEGER PRIMARY KEY AUTOINCREMENT,
13               company TEXT,
14               amount REAL,
15               payment_date TEXT,
16               status TEXT,
17               due_date TEXT,
18               tax_rate REAL)')
19    conn.commit()
20    conn.close()
21
22 create_table()
```

- **/ (GET)** : Fetches all payment records and displays them on the main page with the dynamic due date drop down

```
Structure supported by PyCharm Professional
23 @app.template_filter('format_date')
24 def format_date(value):
25     if value:
26         return datetime.datetime.strptime(value, format='%Y-%m-%d').strftime('%m/%d/%Y')
27     return ''
28
29 @app.route('/')
30 def index():
31     conn = sqlite3.connect(DATABASE)
32     c = conn.cursor()
33
34     current_year = datetime.datetime.now().year
35
36     next_year = current_year + 1
37     due_dates = [
38         f"04/15/{current_year}",
39         f"06/15/{current_year}",
40         f"09/15/{current_year}",
41         f"01/15/{next_year}"
42     ]
43     c.execute('SELECT * FROM payments')
44     records = c.fetchall()
45     print(records)
46     return render_template(template_name_or_list='index.html', due_dates=due_dates, records=records)
47
```

- **/submit (POST)** : Adds a new payment record to the database using data submitted via html form.

```
flask is supported by PyCharm Professional
48 @app.template_filter('datetimeformat')
49 def datetimeformat(value, format='%Y-%m-%d'):
50     if isinstance(value, datetime.datetime):
51         return value.strftime(format)
52     else:
53         return value
54
55 @app.route(rule='/submit', methods=['POST']) 2 usages (2 dynamic)
56 def submit():
57     company = request.form['company']
58     amount = float(request.form['amount'])
59     payment_date = request.form['paymentDate']
60     status = request.form['status']
61     due_date = request.form['dueDate']
62     tax_rate = float(request.form['taxRate'])
63     conn = sqlite3.connect(DATABASE)
64     c = conn.cursor()
65     c.execute(sql='INSERT INTO payments (company, amount, payment_date, status, due_date, tax_rate)
66             VALUES (?, ?, ?, ?, ?, ?)', parameters=(company, amount, payment_date, status, due_date, tax_rate))
67     conn.commit()
68     conn.close()
69     return redirect(url_for('index'))
70
```

- **/insert (POST)** : Used for inserting new records into the database.

```
flask is supported by PyCharm Professional

70
71 @app.route(rule='/insert', methods=['POST'])
72 def insert_record():
73     data = request.get_json()
74     conn = sqlite3.connect(DATABASE)
75     c = conn.cursor()
76     c.execute(sql: '''INSERT INTO payments (company, amount, payment_date, status, due_date, tax_rate)
77         VALUES (?, ?, ?, ?, ?, ?)''',
78         parameters: (data['company'], data['amount'], data['payment_date'],
79             data['status'], data['due_date'], data['tax_rate']))
80     conn.commit()
81     conn.close()
82     return 'Record inserted successfully'
83
84 def format_date_mmdyyy(date_str): 1 usage
85     if date_str:
86         parts = date_str.split('-')
87         if len(parts) == 3:
88             return f"{parts[1]}/{parts[2]}/{parts[0]}"
89     return date_str
```

- **/summary (GET)** : Returns filtered payment records and calculated totals based on the selected due date.

```
flask is supported by PyCharm Professional

91 @app.route('/summary')
92 def summary():
93     due_date = request.args.get('dueDate')
94     conn = sqlite3.connect(DATABASE)
95     c = conn.cursor()
96     c.execute(sql: '''SELECT * FROM payments WHERE due_date=?''', parameters: (due_date,))
97     data = c.fetchall()
98     total_amount = sum(row[2] for row in data)
99     tax_rate = data[0][6] if data else 0
100     tax_due = total_amount * tax_rate
101     conn.close()
102     html_parts = [
103         '<table border="1">',
104         '<tr><th>ID</th><th>Company</th><th>Amount</th><th>Payment Dates</th><th>Status</th><th>Due Date</th></tr>'
105     ]
106     for row in data:
107         payment_date = row[3] if row[3] else 'NA'
108         formatted_date = format_date_mmdyyy(payment_date)
109         html_parts.append(
110             f'<tr><td>{row[0]}</td><td>{row[1]}</td><td>{row[2]}</td><td>{formatted_date}</td><td>{row[4]}</td><td>{row[5]}</td></tr>')
111     html_parts.append(f'<tr><td colspan="5"><strong>Total Amount:</strong></td><td>&dollar;{total_amount}</td></tr>')
112     html_parts.append(f'<tr><td colspan="5"><strong>Tax Rate:</strong></td><td>{tax_rate * 100}%</td></tr>')
113     html_parts.append(f'<tr><td colspan="5"><strong>Tax Due:</strong></td><td>&dollar;{tax_due}</td></tr>')
114     html_parts.append('</table>')
115     html = ''.join(html_parts)
116     return jsonify({'html': html})
```

- **/update (POST)** : Updates the existing payment record based on the inserted new data.

```

flask is supported by PyCharm Professional

119 @app.route(rule: '/update', methods=['POST'])
120 def update():
121     edit_id = request.form['editId']
122     company = request.form['editCompany']
123     amount = float(request.form['editAmount'])
124     payment_date = request.form['editPaymentDate']
125     status = request.form['editStatus']
126     due_date = request.form['editDueDate']
127     tax_rate = float(request.form['editTaxRate'])
128
129     conn = sqlite3.connect(DATABASE)
130     c = conn.cursor()
131     c.execute(sql: 'UPDATE payments SET company=?, amount=?, payment_date=?, status=?, due_date=?, tax_rate=? WHERE id=?',
132             parameters: (company, amount, payment_date, status, due_date, tax_rate, edit_id))
133     conn.commit()
134     conn.close()
135
136     return jsonify({'success': True})

```

- **/delete (DELETE)** : Deletes a specific payment record by indentifying id.

```

137
138 @app.route(rule: '/delete', methods=['DELETE'])
139 def delete():
140     delete_id = request.args.get('id')
141     conn = sqlite3.connect(DATABASE)
142     c = conn.cursor()
143     c.execute(sql: 'DELETE FROM payments WHERE id=?', parameters: (delete_id,))
144     conn.commit()
145     c.execute('SELECT MAX(id) FROM payments')
146     max_id = c.fetchone()[0]
147     if max_id is None:
148         max_id = 0
149     c.execute(sql: 'UPDATE sqlite_sequence SET seq = ? WHERE name = "payments"', parameters: (max_id,))
150     conn.commit()
151     conn.close()
152     return jsonify({'success': True})
153
154 if __name__ == '__main__':
155     app.run(debug=True)

```

## Running the code

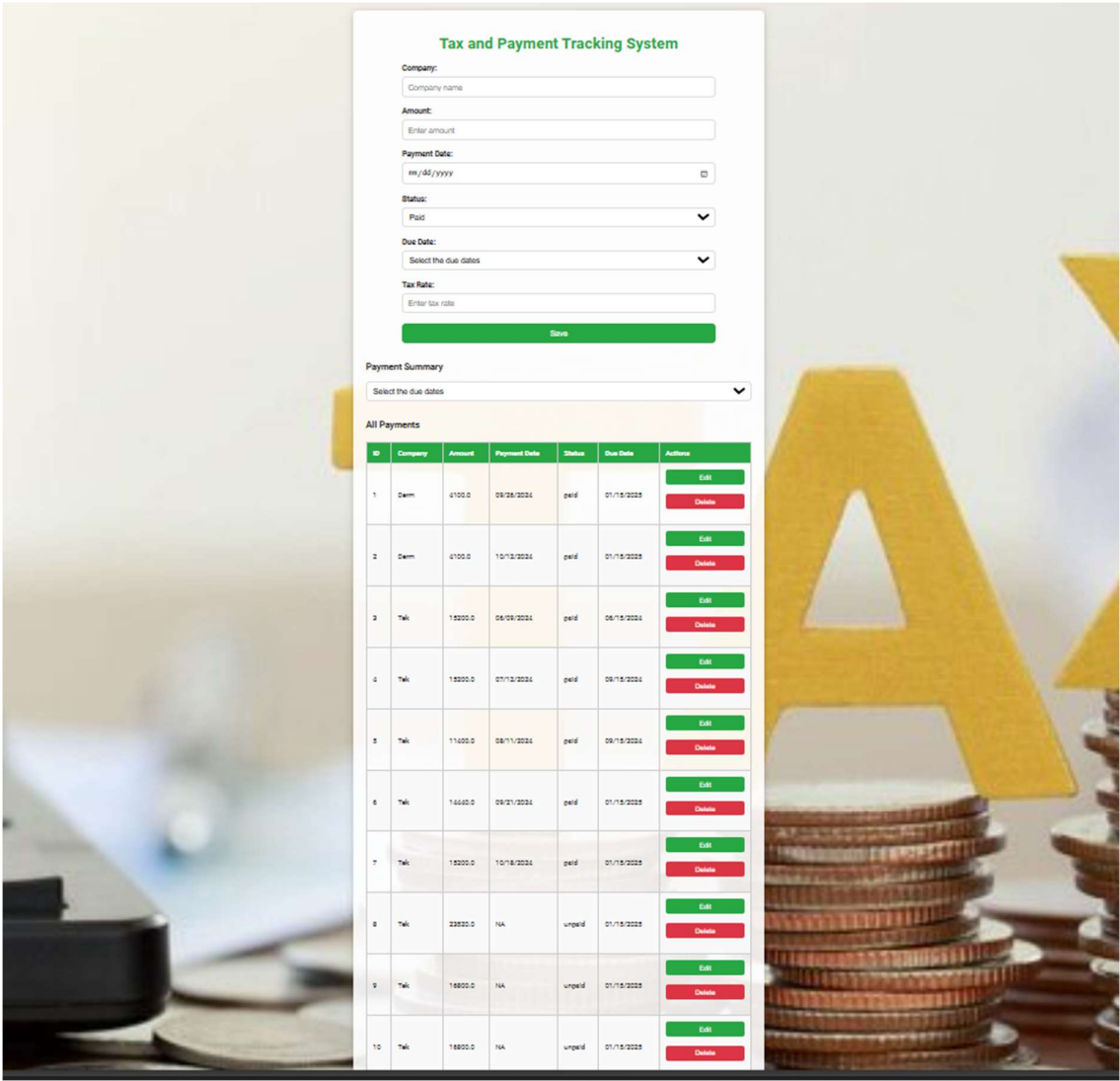
```

(.venv) PS C:\Users\Aaryaman\PycharmProjects\Aaryaman_Singh_Patel_Final_Project_CSIT555> python app.py
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 478-312-323

```

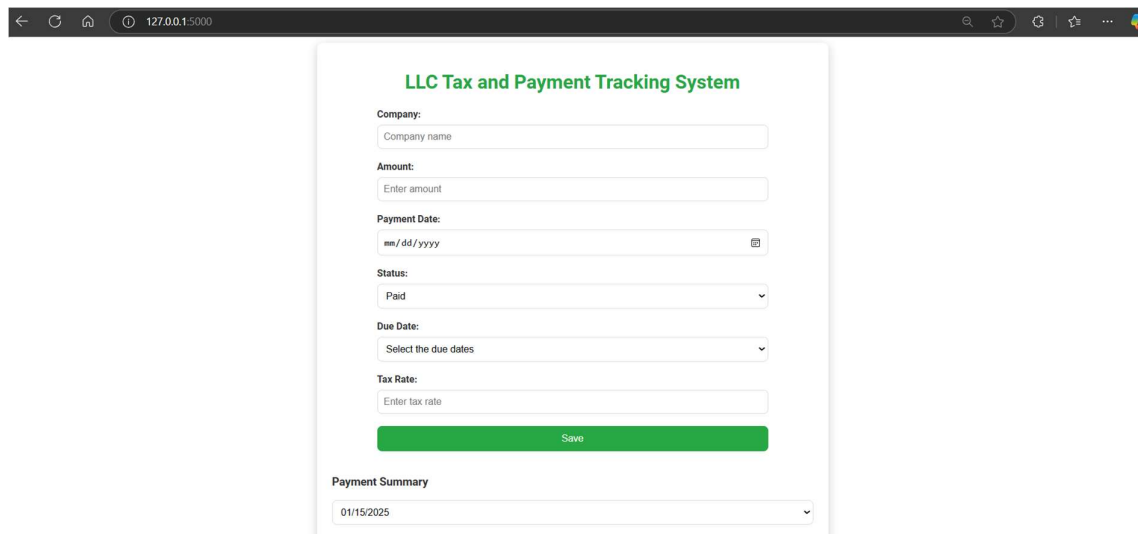


# UI and WORKFLOW OF THE WEB APPLICATION





## 1. Details section

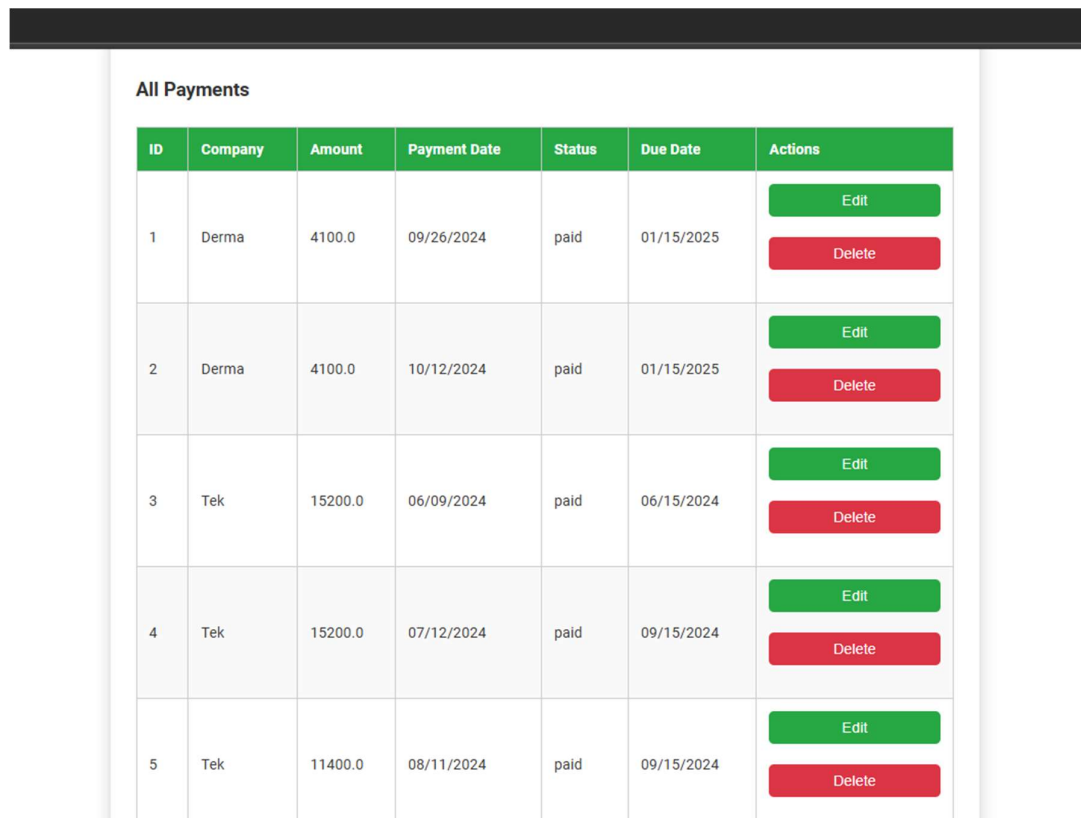


The screenshot shows a web browser window with the address bar displaying "127.0.0.1:5000". The page title is "LLC Tax and Payment Tracking System". The form contains the following fields:

- Company:** A text input field labeled "Company name".
- Amount:** A text input field labeled "Enter amount".
- Payment Date:** A date input field labeled "mm/dd/yyyy" with a calendar icon.
- Status:** A dropdown menu with "Paid" selected.
- Due Date:** A dropdown menu labeled "Select the due dates".
- Tax Rate:** A text input field labeled "Enter tax rate".
- Save:** A green button.
- Payment Summary:** A dropdown menu showing "01/15/2025".

We fill in the details like company name, amount, payment date, status, due date, tax rate. Then the data is stored and show in the All payments section.

## 2. All Payment Section

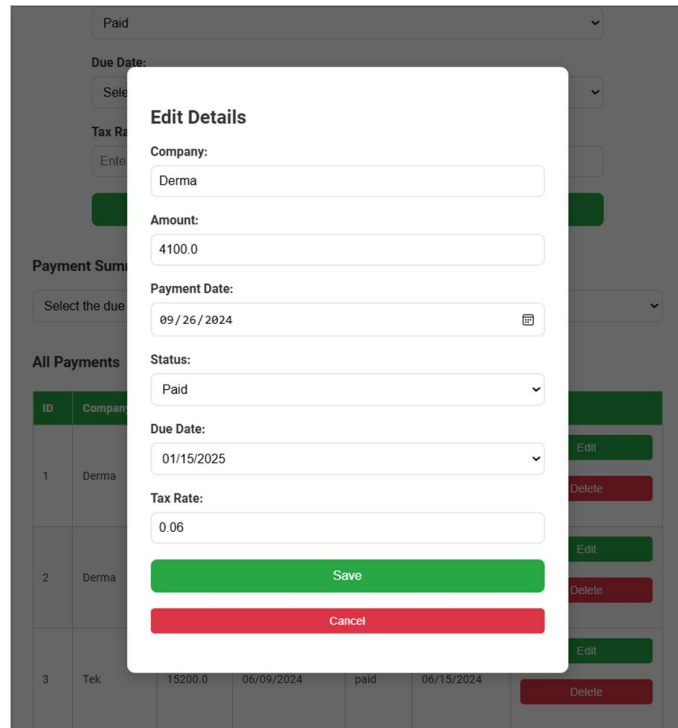


The screenshot shows a web browser window with a dark header bar. Below the header, the section is titled "All Payments". It contains a table with the following data:

ID	Company	Amount	Payment Date	Status	Due Date	Actions
1	Derma	4100.0	09/26/2024	paid	01/15/2025	<div>Edit</div> <div>Delete</div>
2	Derma	4100.0	10/12/2024	paid	01/15/2025	<div>Edit</div> <div>Delete</div>
3	Tek	15200.0	06/09/2024	paid	06/15/2024	<div>Edit</div> <div>Delete</div>
4	Tek	15200.0	07/12/2024	paid	09/15/2024	<div>Edit</div> <div>Delete</div>
5	Tek	11400.0	08/11/2024	paid	09/15/2024	<div>Edit</div> <div>Delete</div>

After the data is stored. We see two buttons in each row stating edit and delete options. When the edit button is clicked a pop up appears where we can edit the details of that particular row.

### 3. POPUP



The image shows a web application interface with a table of payments and an 'Edit Details' popup form. The table has columns for ID, Company, Amount, Due Date, Status, and Action. The popup form allows editing the details of a selected payment record.

ID	Company	Amount	Due Date	Status	Action
1	Derma	4100.0	01/15/2025	Paid	Edit Delete
2	Derma				Edit Delete
3	Tek	15200.0	06/09/2024	paid	06/15/2024 Edit Delete

#### Edit Details

Company:

Amount:

Payment Date:

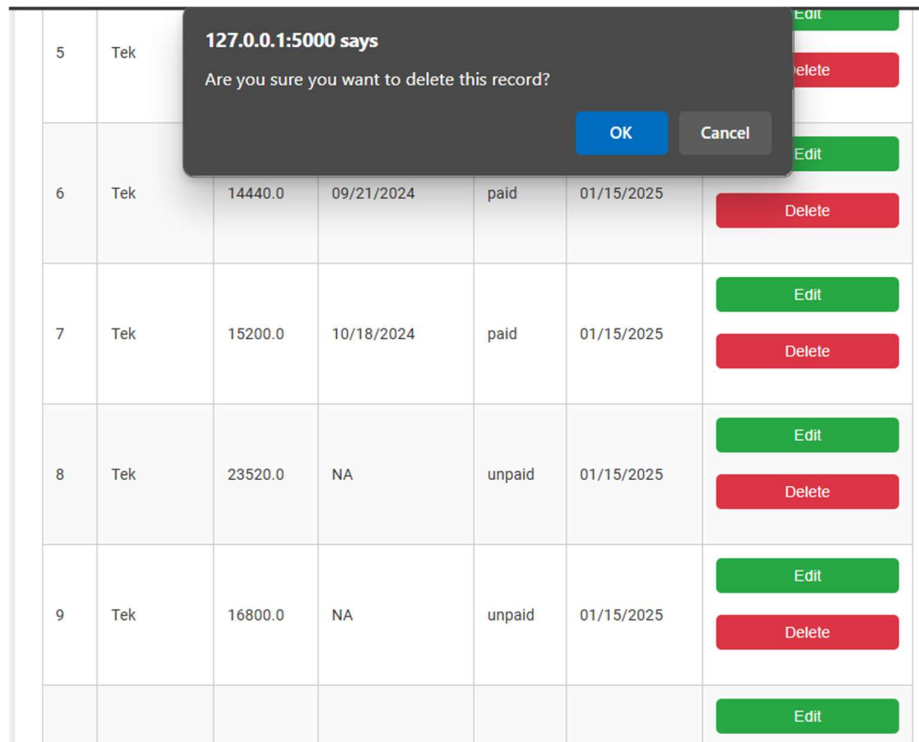
Status:

Due Date:

Tax Rate:

Here after clicking edit a popup appears and we can edit the record and save it by clicking the save button or just cancel it if there is a change of mind.

#### 4. Delete Confirmation POPUP

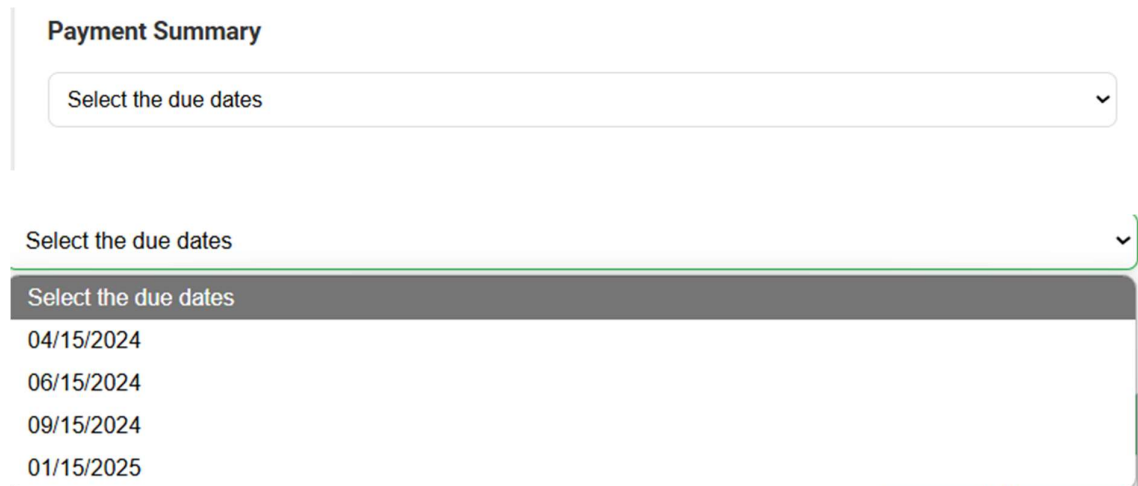


The image shows a table with 6 columns and 6 rows. A dark gray popup is centered over the table, displaying the text "127.0.0.1:5000 says" and "Are you sure you want to delete this record?". The popup has two buttons: "OK" (blue) and "Cancel" (gray). The table contains the following data:

5	Tek					<div>Edit</div> <div>Delete</div>
6	Tek	14440.0	09/21/2024	paid	01/15/2025	<div>Edit</div> <div>Delete</div>
7	Tek	15200.0	10/18/2024	paid	01/15/2025	<div>Edit</div> <div>Delete</div>
8	Tek	23520.0	NA	unpaid	01/15/2025	<div>Edit</div> <div>Delete</div>
9	Tek	16800.0	NA	unpaid	01/15/2025	<div>Edit</div> <div>Delete</div>
						<div>Edit</div>

When trying to delete a record by clicking a delete button we are shown a popup button as shown in the image above.

#### 5. PAYMENT SUMMARY SECTION



The image shows a "Payment Summary" section. It contains a dropdown menu labeled "Select the due dates". The dropdown is open, showing a list of dates: 04/15/2024, 06/15/2024, 09/15/2024, and 01/15/2025.

### Payment Summary

04/15/2024

ID	Company	Amount	Payment Dates	Status	Due Date
Total Amount:					\$0
Tax Rate:					0%
Tax Due:					\$0

### Payment Summary

09/15/2024

ID	Company	Amount	Payment Dates	Status	Due Date
4	Tek	15200.0	07/12/2024	paid	09/15/2024
5	Tek	11400.0	08/11/2024	paid	09/15/2024
Total Amount:					\$26600.0
Tax Rate:					6.0%
Tax Due:					\$1596.0

In the payment summary section, a due date drop down is created by clicking on the drop down, We can select one particular due date that provides the calculated tax due amount of that date. The details we see in the table are total amount, tax rate and tax due.

## VIDEO AND GITHUB LINKS

[https://youtu.be/5Iz89Q\\_Zk3c](https://youtu.be/5Iz89Q_Zk3c)

[https://github.com/Aaryaman-Singh-Patel/Aaryaman\\_Singh\\_Patel\\_Final\\_Project-CSIT555\\_01FA24.git](https://github.com/Aaryaman-Singh-Patel/Aaryaman_Singh_Patel_Final_Project-CSIT555_01FA24.git)