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

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

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Tax and Payment Tracking System</title>
    <link rel="stylesheet" href="/static/style.css">
</head>
<body>
    <div class="container">
        <h1>LLC Tax and Payment Tracking System</h1>
        <form id="taxForm" action="/submit" method="POST">
            <label for="company">Company:</label>
            <input type="text" id="company" name="company" placeholder="Company name" required>
            <label for="amount">Amount:</label>
            <input type="number" id="amount" name="amount" step="0.01" placeholder="Enter amount" required>
            <label for="paymentDate">Payment Date:</label>
            <input type="date" id="paymentDate" name="paymentDate">
            <label for="status">Status:</label>
            <select id="status" name="status" required>
                <option value="paid">Paid</option>
                <option value="unpaid">Unpaid</option>
            </select>
            <label for="dueDate">Due Date:</label>
            <select id="dueDate" name="dueDate" required>
                <option value="">Select the due dates</option>
                {% for due_date in due_dates %}
                <option value="{{ due_date }}">{{ due_date }}</option>
                {% endfor %}
            </select>
```

```html
            <label for="taxRate">Tax Rate:</label>
            <input type="number" id="taxRate" name="taxRate" step="0.01" placeholder="Enter tax rate" required>
            <button type="submit">Save</button>
        </form>

        <h3>Payment Summary</h3>
        <select id="summaryDueDate">
            <option value="">Select the due dates</option>
            {% for due_date in due_dates %}
            <option value="{{ due_date }}">{{ due_date }}</option>
            {% endfor %}
        </select>
        <div id="summary"></div>

        <h3>All Payments</h3>
        <table class="table table-striped">
            <thead>
                <tr>
                    <th>ID</th>
                    <th>Company</th>
                    <th>Amount</th>
                    <th>Payment Date</th>
                    <th>Status</th>
                    <th>Due Date</th>
                    <th>Actions</th>
                </tr>
            </thead>
```

```html
            <tbody>
                {% for records in records %}
                <tr>
                    <td>{{ records[0] }}</td>
                    <td>{{ records[1] }}</td>
                    <td>{{ records[2] }}</td>
                    <td>{% if records[3] %}{{ records[3]|format_date }}{% else %}NA{% endif %}</td>
                    <td>{{ records[4] }}</td>
                    <td>{{ records[5] }}</td>
                    <td>
                        <button class="edit-btn" onclick="openEditPopup('{{ records[0] }}', '{{ records[1] }}', '{{ records[2] }}',
                        SS '{{ records[3] }}', '{{ records[4] }}', '{{ records[5] }}', '{{ records[6] }}')">Edit</button>
                        <button class="delete-btn" onclick="deleteRecord('{{ records[0] }}')">Delete</button>
                    </td>
                </tr>
                {% endfor %}
            </tbody>
        </table>
    </div>
```

```html
<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>
```

```html
    <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.

```css
body {
    font-family: 'Roboto', Arial, sans-serif;
    background-image: url('bg_tax.jpg');
    background-size: cover;
    background-repeat: no-repeat;
    background-position: center;
    margin: 0;
    padding: 0;
    color: #333;
}

.container {
    max-width: 800px;
    margin: 20px auto;
    padding: 25px;
    background-color: rgba(255, 255, 255, 0.9);
    box-shadow: 0 4px 15px rgba(0, 0, 0, 0.2);
    border-radius: 10px;
}

h1 {
    text-align: center;
    margin-bottom: 25px;
    font-size: 2rem;
    color: #28a745;
}

form {
    max-width: 650px;
    margin: 0 auto;
```

```css
}

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);
}
```

```css
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;
```

```css
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.

```javascript
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';
}
```

```javascript
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.
     i. @app.route('/submit'): This route handles the insertion of new payment records.
     ii. @app.route('/update'): This route updates existing records.
     iii. @app.route('/delete'): This route deletes records based on the user's action.
     iv. @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.

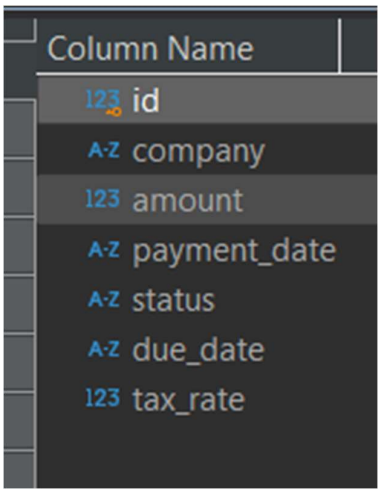
3. **Database Implementation:**
   - **SQLite:**
     i. The database is used for storing payment details. The schema includes fields like id, company, amount, payment_date, status, due_date and tax_rate.
     ii. CRUD operations are performed with the help of SQL queries which are written in the Flask 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 un der 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 yyy-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.



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)

flask is supported by PyCharm Professional Try PyCharm Professional Dismiss

```python
from flask import Flask, render_template, request, redirect, url_for, jsonify
import sqlite3
import datetime

app = Flask(__name__)
DATABASE = 'database.db'

def create_table():  1 usage
    conn = sqlite3.connect(DATABASE)
    c = conn.cursor()
    c.execute('''CREATE TABLE IF NOT EXISTS payments
                 (id INTEGER PRIMARY KEY AUTOINCREMENT,
                  company TEXT,
                  amount REAL,
                  payment_date TEXT,
                  status TEXT,
                  due_date TEXT,
                  tax_rate REAL)''')
    conn.commit()
    conn.close()

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
@app.template_filter('format_date')
def format_date(value):
    if value:
        return datetime.datetime.strptime(value, format: '%Y-%m-%d').strftime('%m/%d/%Y')
    return ''

@app.route('/')
def index():
    conn = sqlite3.connect(DATABASE)
    c = conn.cursor()

    current_year = datetime.datetime.now().year

    next_year = current_year + 1
    due_dates = [
        f"04/15/{current_year}",
        f"06/15/{current_year}",
        f"09/15/{current_year}",
        f"01/15/{next_year}"
    ]
    c.execute('''SELECT * FROM payments''')
    records = c.fetchall()
    print(records);
    return render_template( template_name_or_list: 'index.html',due_dates=due_dates,records=records)
```

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

```
flask is supported by PyCharm Professional
@app.template_filter('datetimeformat')
def datetimeformat(value, format='%Y-%m-%d'):
    if isinstance(value, datetime.datetime):
        return value.strftime(format)
    else:
        return value

@app.route( rule: '/submit', methods=['POST'])  2 usages (2 dynamic)
def submit():
    company = request.form['company']
    amount = float(request.form['amount'])
    payment_date = request.form['paymentDate']
    status = request.form['status']
    due_date = request.form['dueDate']
    tax_rate = float(request.form['taxRate'])
    conn = sqlite3.connect(DATABASE)
    c = conn.cursor()
    c.execute( sql: '''INSERT INTO payments (company, amount, payment_date, status, due_date, tax_rate)
                VALUES (?, ?, ?, ?, ?, ?)''',  parameters: (company, amount, payment_date, status, due_date, tax_rate))
    conn.commit()
    conn.close()
    return redirect(url_for('index'))
```

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

flask is supported by PyCharm Professional

```python
@app.route( rule: '/insert', methods=['POST'])
def insert_record():
    data = request.get_json()
    conn = sqlite3.connect(DATABASE)
    c = conn.cursor()
    c.execute( sql: '''INSERT INTO payments (company, amount, payment_date, status, due_date, tax_rate)
                VALUES (?, ?, ?, ?, ?, ?)''',
               parameters: (data['company'], data['amount'], data['payment_date'],
                data['status'], data['due_date'], data['tax_rate']))
    conn.commit()
    conn.close()
    return 'Record inserted successfully'

def format_date_mmddyyyy(date_str):  1 usage
    if date_str:
        parts = date_str.split('-')
        if len(parts) == 3:
            return f"{parts[1]}/{parts[2]}/{parts[0]}"
    return date_str
```

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

flask is supported by PyCharm Professional

```python
@app.route('/summary')
def summary():
    due_date = request.args.get('dueDate')
    conn = sqlite3.connect(DATABASE)
    c = conn.cursor()
    c.execute( sql: '''SELECT * FROM payments WHERE due_date=?''', parameters: (due_date,))
    data = c.fetchall()
    total_amount = sum(row[2] for row in data)
    tax_rate = data[0][6] if data else 0
    tax_due = total_amount * tax_rate
    conn.close()
    html_parts = [
        '<table border="1">',
        '<tr><th>ID</th><th>Company</th><th>Amount</th><th>Payment Dates</th><th>Status</th><th>Due Date</th></tr>'
    ]
    for row in data:
        payment_date = row[3] if row[3] else 'NA'
        formatted_date = format_date_mmddyyyy(payment_date)
        html_parts.append(
            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>')
    html_parts.append(f'<tr><td colspan="5"><strong>Total Amount:</strong></td><td>&dollar;{total_amount}</td></tr>')
    html_parts.append(f'<tr><td colspan="5"><strong>Tax Rate:</strong></td><td>{tax_rate * 100}%</td></tr>')
    html_parts.append(f'<tr><td colspan="5"><strong>Tax Due:</strong></td><td>&dollar;{tax_due}</td></tr>')
    html_parts.append('</table>')
    html = ''.join(html_parts)
    return jsonify({'html': html})
```

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

```
flask is supported by PyCharm Professional
@app.route( rule: '/update', methods=['POST'])
def update():
    edit_id = request.form['editId']
    company = request.form['editCompany']
    amount = float(request.form['editAmount'])
    payment_date = request.form['editPaymentDate']
    status = request.form['editStatus']
    due_date = request.form['editDueDate']
    tax_rate = float(request.form['editTaxRate'])

    conn = sqlite3.connect(DATABASE)
    c = conn.cursor()
    c.execute( sql: '''UPDATE payments SET company=?, amount=?, payment_date=?, status=?, due_date=?, tax_rate=? WHERE id=?''',
               parameters: (company, amount, payment_date, status, due_date, tax_rate, edit_id))
    conn.commit()
    conn.close()

    return jsonify({'success': True})
```

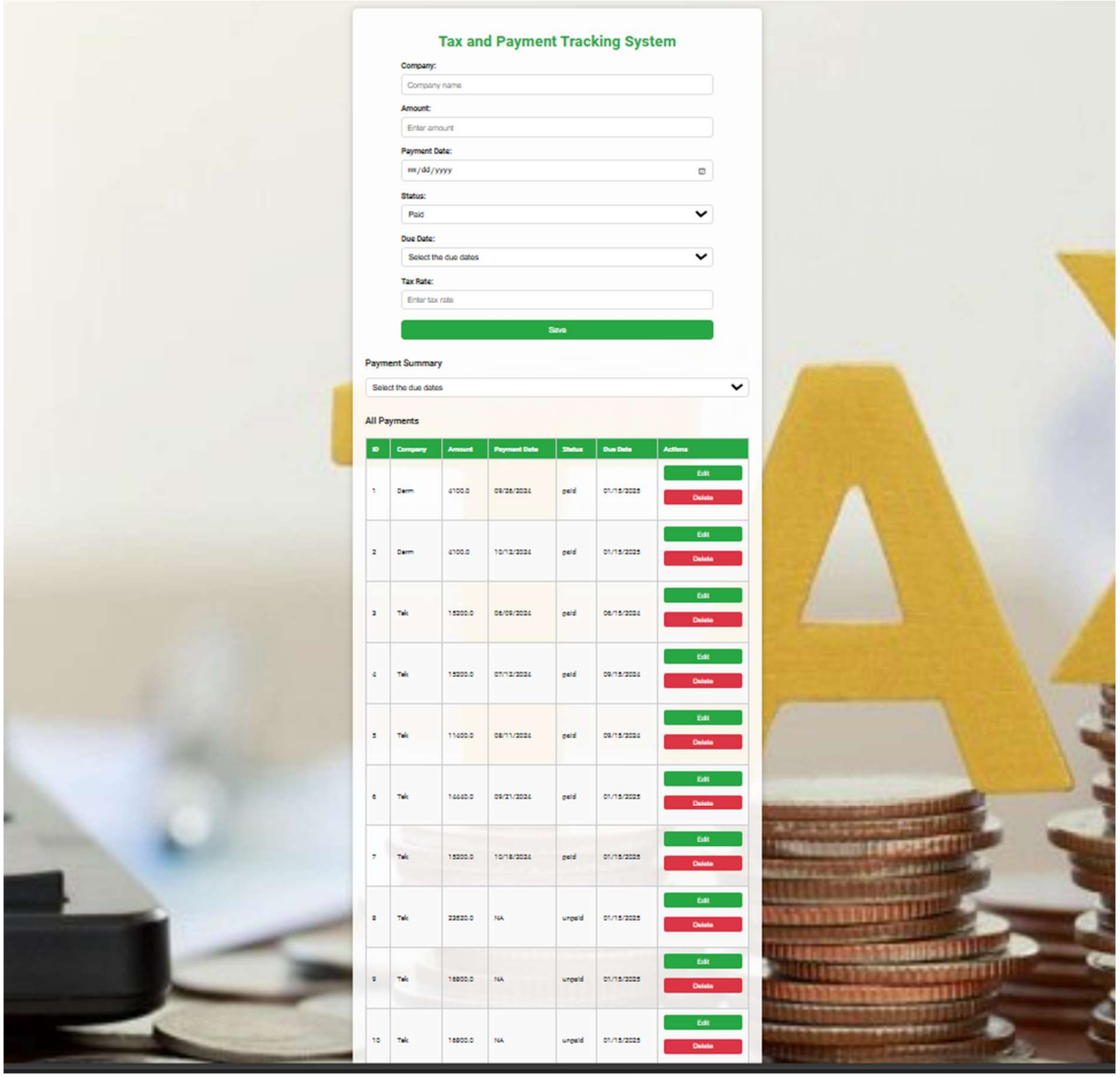
- **/delete (DELETE) :** Deletes a specific payment record by indentifying id.

```

@app.route( rule: '/delete', methods=['DELETE'])
def delete():
    delete_id = request.args.get('id')
    conn = sqlite3.connect(DATABASE)
    c = conn.cursor()
    c.execute( sql: 'DELETE FROM payments WHERE id=?', parameters: (delete_id,))
    conn.commit()
    c.execute('SELECT MAX(id) FROM payments')
    max_id = c.fetchone()[0]
    if max_id is None:
        max_id = 0
    c.execute( sql: 'UPDATE sqlite_sequence SET seq = ? WHERE name = "payments"', parameters: (max_id,))
    conn.commit()
    conn.close()
    return jsonify({'success': True})

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

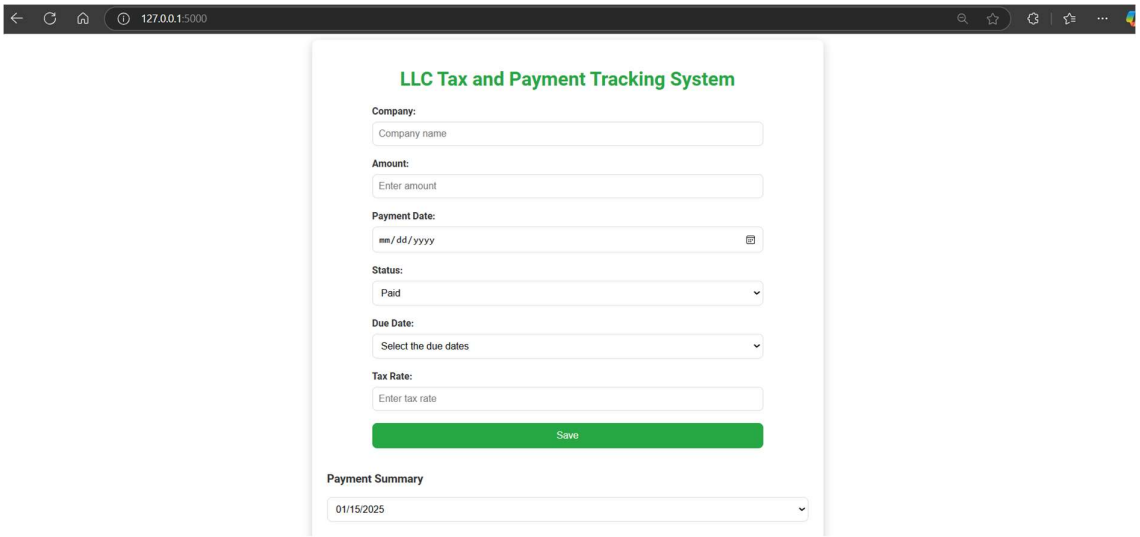
Running the code

```
(.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

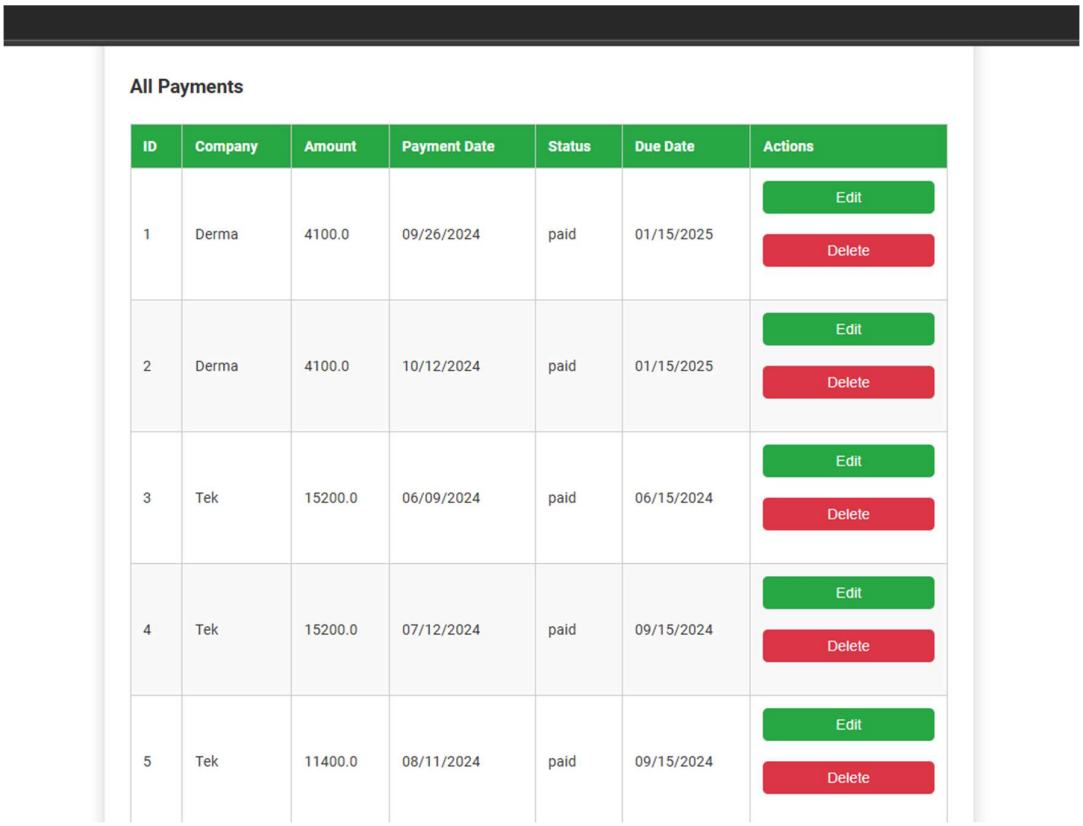


1. Details section



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

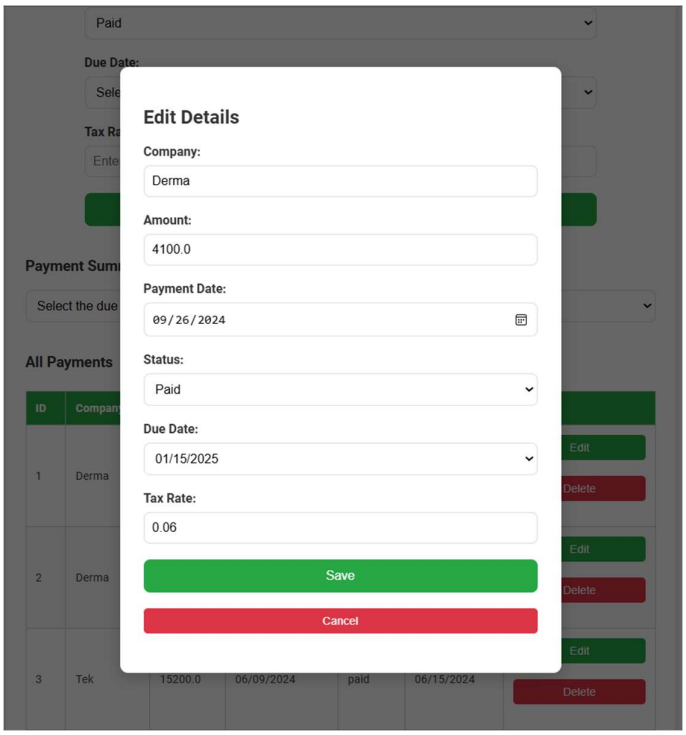


All Payments

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

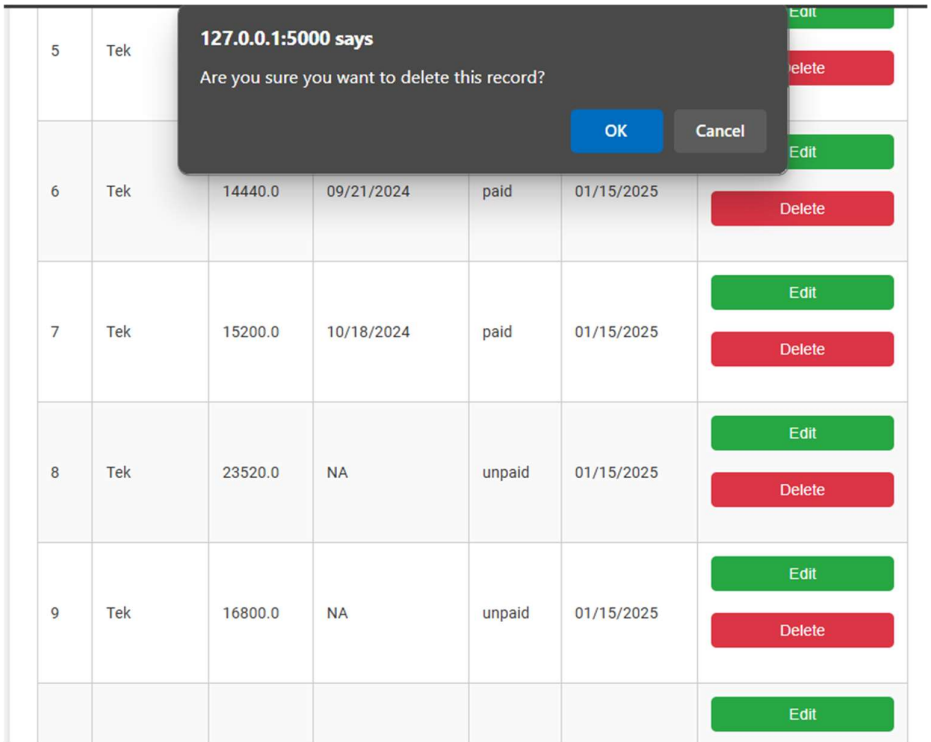
After the data is stored. We see two buttons in each row stating edit and delete options. When the edit button is clicked a pop up appears where we can edit the details of that particular row.

3. POPUP



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



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

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://github.com/Aaryaman-Singh-Patel/Aaryaman_Singh_Patel_Final_Project-CSIT555_01FA24.git