

**DATABASE SYSTEMS – FINAL PROJECT**

**CSIT 555**

**SUBMITTED TO**

**INSTRUCTOR: XIAOFENG LI (KEVIN)**

**MONTCLAIR STATE UNIVERSITY**

**SUBMITTED BY :**

**GAURI PRAKASH DHANAWADE**

**TABLE OF CONTENT**

**Introduction 3**

**System Architecture 3-4**

**Database Structure and Code 4-5**

**Database Connection 6**

**SQL Schema 6**

**Controller and Endpoints 6-8**

**API’s 8-9**

**Screenshots 9-12**

**Git Repository Link 13**

**YouTube Video Link 13**

**INTRODUCTION**

The Tax and Payment Tracking System is an online application that helps businesses and individuals handle their tax payments more efficiently. This system, built using Flask, SQLite, and JavaScript, provides an easy-to-use interface for effectively recording, monitoring, and analysing taxes. This system allows users to conveniently enter data about their tax payments, such as the firm name, payment amount, payment date, status, due date, and tax rate. The application allows users to add, update, and delete payment records, giving them greater flexibility and control over their tax data.

**SYSTEM ARCHITECTURE**

The Tax and Payment Tracking System is a web-based application built using Flask as the web framework, SQLite as the relational database management system, and JavaScript for dynamic client-side interactions. The system has a three-tier architecture, which includes the presentation layer, the application layer, and the data layer, as stated below:

**Presentation Layer**

* The presentation layer handles the application's user interface (UI) components.
* It is made up of HTML templates, CSS stylesheets, and client-side JavaScript code.
* HTML templates are used to structure web page layouts and show dynamic content via template engines such as Jinja2.
* CSS stylesheets define the visual appearance and layout of UI elements, resulting in a consistent and visually pleasing user experience.
* Client-side JavaScript code improves interactivity and dynamic behaviour, including form validation, AJAX queries, and DOM manipulation.
* Input forms, tables, buttons, and modal dialogs are examples of UI components that offer users a responsive and intuitive interface for interacting with the system.

**Application Layer**

* The application layer contains the system's business logic as well as its server-side functions.
* It is built with the Flask web framework, which manages HTTP requests, URL routing, and request handling.
* Flask routes are used to bind URL endpoints to Python functions (view functions) that handle requests, interact with the database, and produce responses.
* View functions use the SQLite3 library to conduct CRUD (Create, Read, Update, and Delete) activities on payment records.
* The application layer also includes AJAX request handlers, which allow the client and server to communicate dynamically and asynchronously.
* In addition, the application layer includes error handling, session management, and authentication procedures to assure security and reliability.

**Data Layer**

* The data layer manages the system's data storage and retrieval.
* It uses SQLite as a relational database management system to store payment records in a structured format.
* The payments table schema specifies fields such as the firm name, payment amount, payment date, status, due date, and tax rate.
* Indexes are established on relevant fields to improve query performance and enable data retrieval.
* CRUD actions on the payments table are carried out with SQL queries conducted in Python using the SQLite3 module.
* Data integrity and consistency are ensured by restrictions such as primary keys, foreign keys, and data type validations.

**DATABASE STRUCTURE AND CODE**

The Tax and Payment Tracking System's database structure consists of a single table named "payments" that records tax payment information. The following SQL code defines the table structure, including primary keys, indexes, and constraints:

**Code:**

def create\_table():

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)''')

# Create indexes

c.execute('''CREATE INDEX IF NOT EXISTS idx\_company ON payments (company)''')

c.execute('''CREATE INDEX IF NOT EXISTS idx\_payment\_date ON payments (payment\_date)''')

c.execute('''CREATE INDEX IF NOT EXISTS idx\_due\_date ON payments (due\_date)''')

c.execute('''CREATE INDEX IF NOT EXISTS idx\_status ON payments (status)''')

c.execute('''CREATE INDEX IF NOT EXISTS idx\_tax\_rate ON payments (tax\_rate)''')

conn.commit()

conn.close()

create\_table()

**Explanation:**

* **ID**: The primary key column has auto-incremental integer values, ensuring that each entry has a unique identity.
* **Company(STRING)**: A column with the name of the firm making the payment. It is defined as NOT NULL to ensure data integrity.
* **Amount(FLOAT)**: A real number column that represents the payment amount. It is defined as NOT NULL with a positive value (CHECK constraint).
* **Payment\_date(STRING (DATE))**: A column containing the date of payment. It supports NULL values for circumstances where the payment date is unclear.
* **Status(STRING)**: A column that indicates the payment status (for example, "paid" or "unpaid"). It's defined as NOT NULL.
* **due\_date(STRING(DATE))**: This column represents the payment's due date. It's defined as NOT NULL.
* **tax\_rate(FLOAT):** A real number column indicating the tax rate applied to the payment amount. It is defined as NOT NULL and must be in the range [0, 1] (CHECK constraint).

**Constraints** are added to ensure data integrity and consistency:

* company\_not\_null: Ensures that the company name is not empty.
* amount\_positive: Ensures that the payment amount is non-negative.
* tax\_rate\_range: Ensures that the tax rate is within the range [0, 1].

**Indexes:**

* Indexes are created on relevant columns to optimize query performance for common search and filter operations.
* Indexes are defined on the "company", "payment\_date", "due\_date", "status", and "tax\_rate" columns to facilitate efficient data retrieval and filtering.

**Primary Key:**

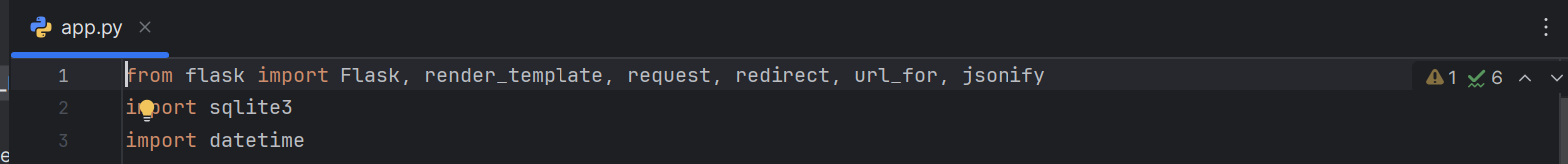
* The primary key is “id INTEGER PRIMARY KEY AUTOINCREMENT”
* id: This column serves as the primary key for the "payments" table.
* INTEGER: Indicates that the data type of the primary key is an integer.
* PRIMARY KEY: Specifies that the "id" column is the primary key, uniquely identifying each record in the table.
* AUTOINCREMENT: This attribute ensures that each new row inserted into the table will automatically be assigned a unique integer value for the "id" column, incrementing from the highest existing value. It prevents the reuse of previously deleted "id" values, ensuring data integrity and uniqueness.

**ER DIAGRAM:**

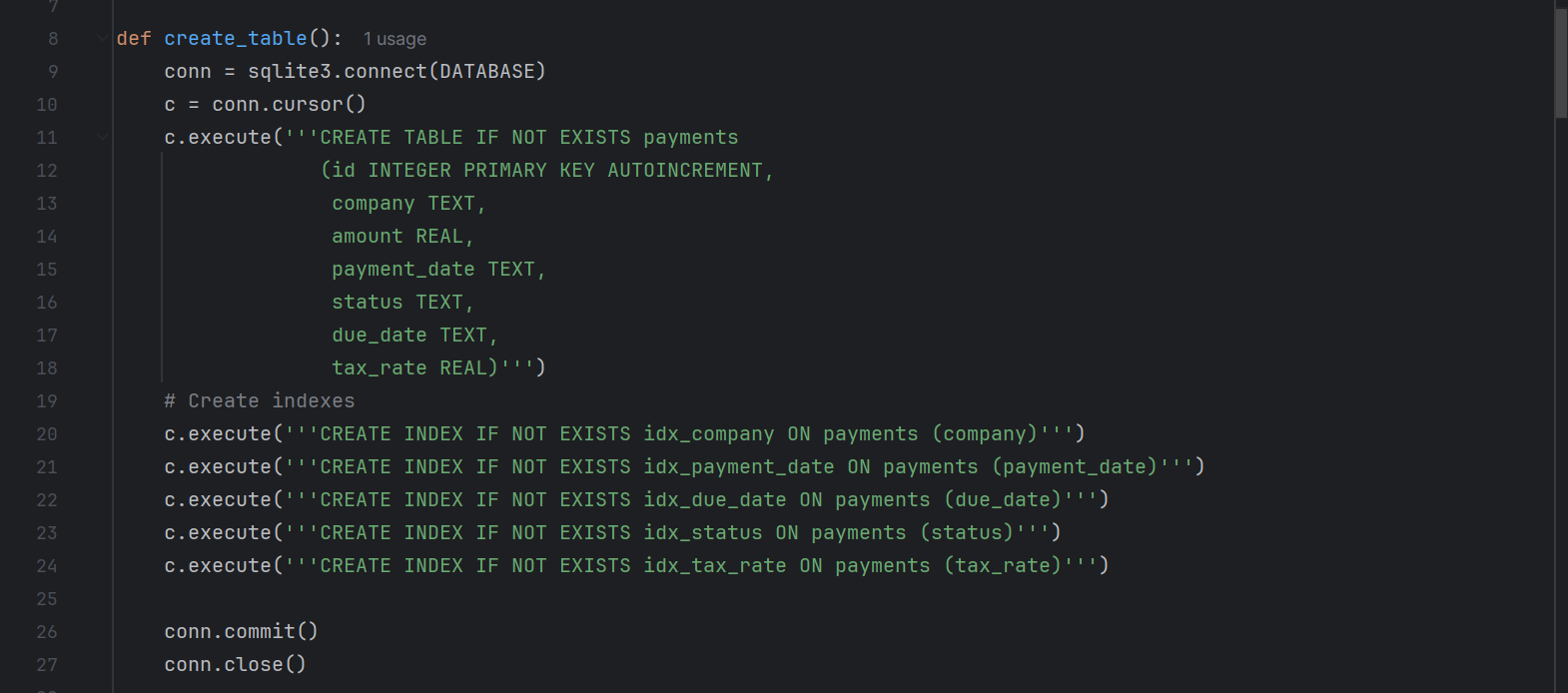
A screenshot of a phone

Description automatically generated

**Database Connection:**

****

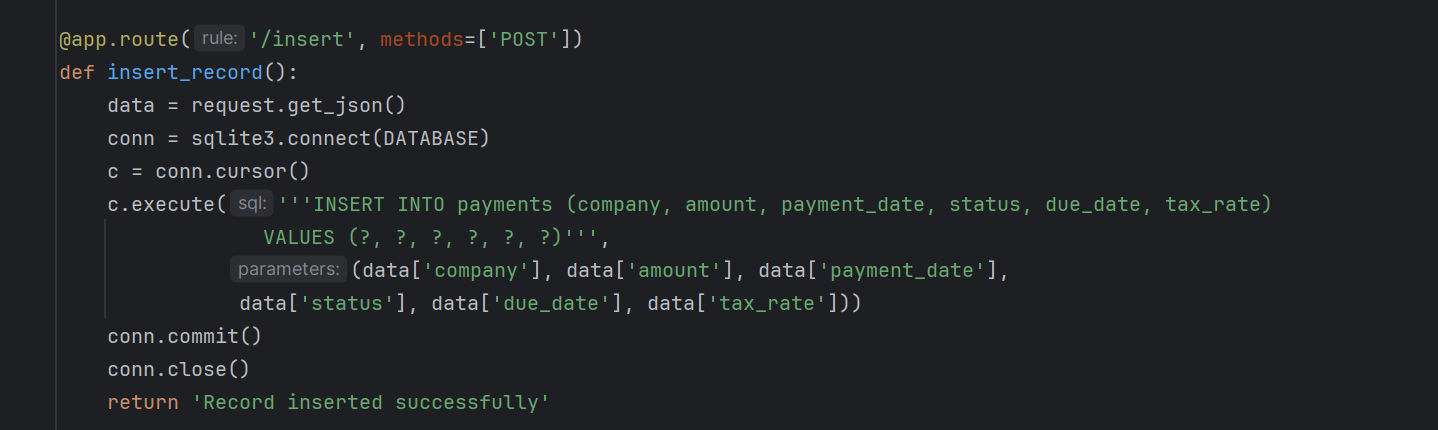
**SQL SCHEMA:**

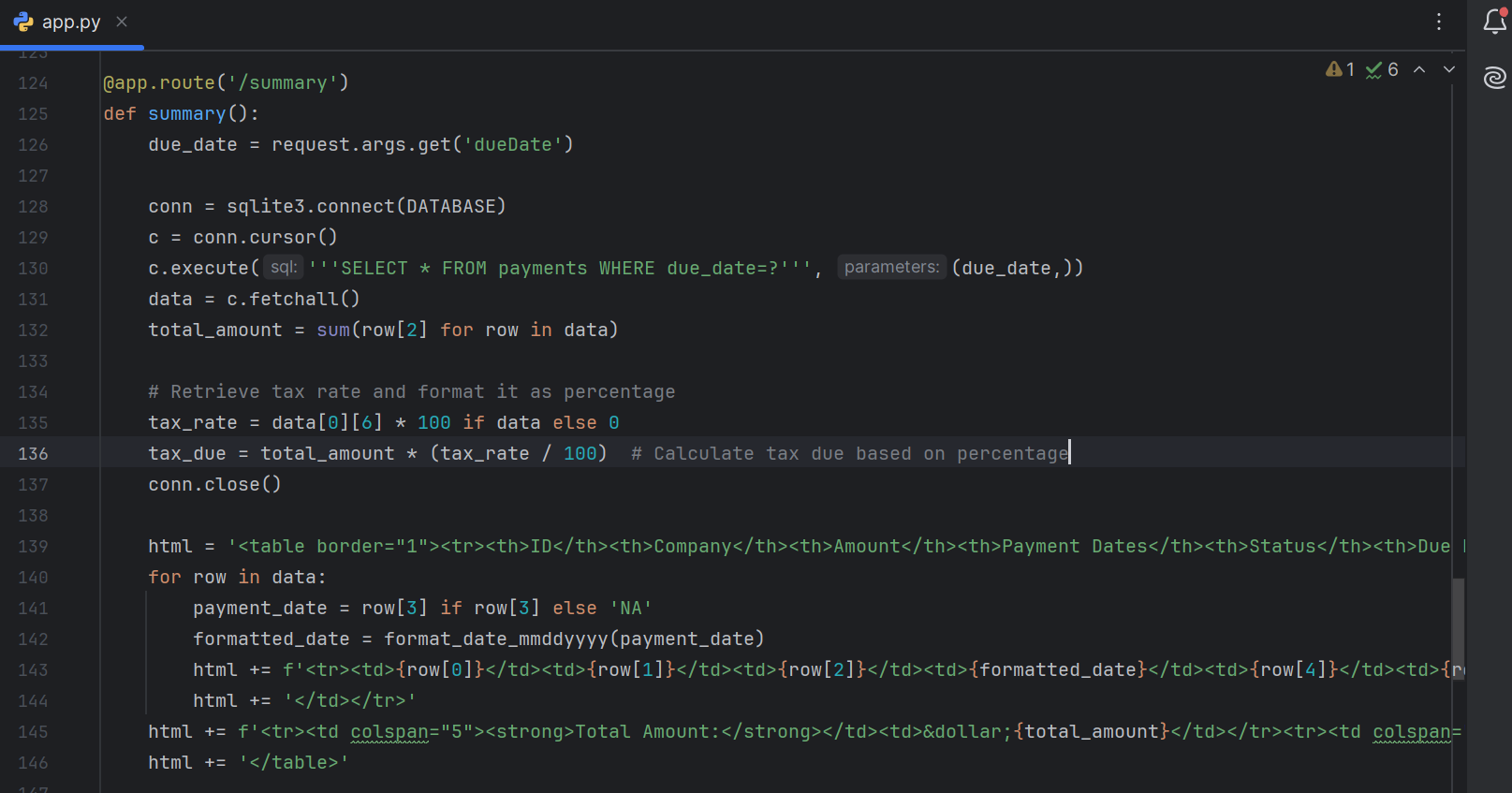
****

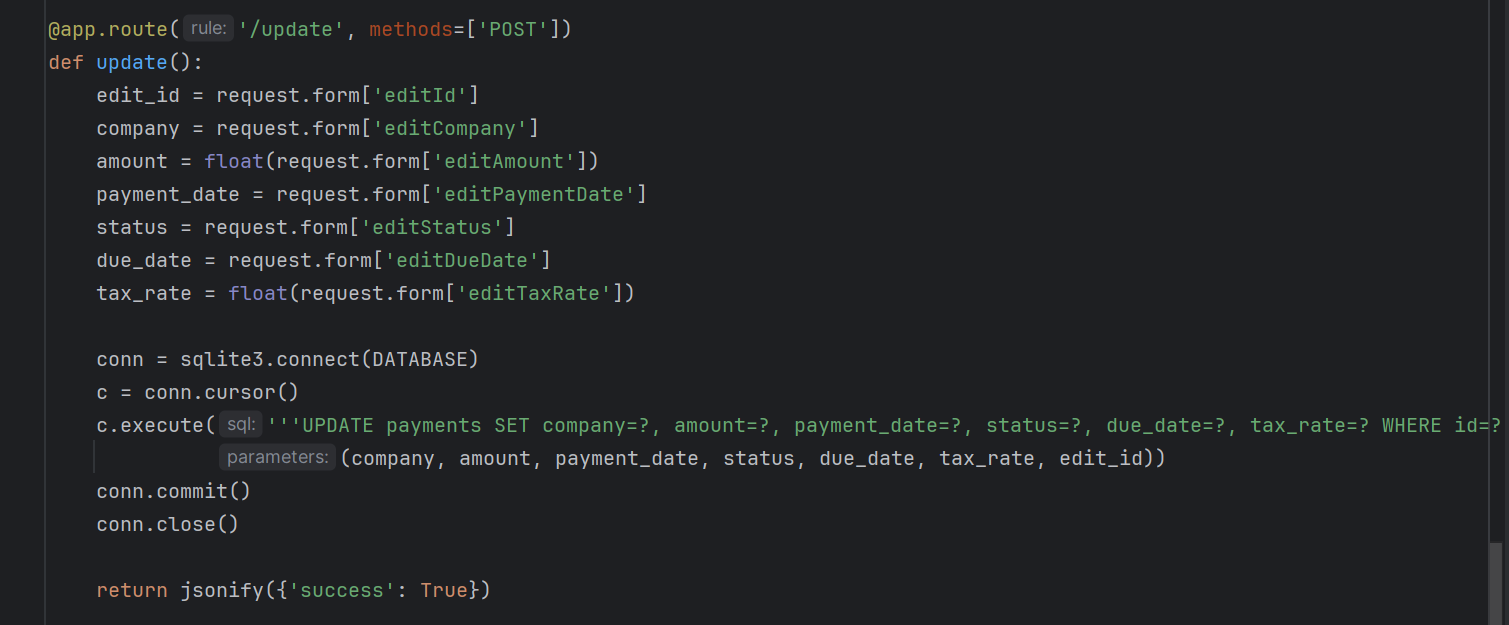
**Controllers and Endpoints:**

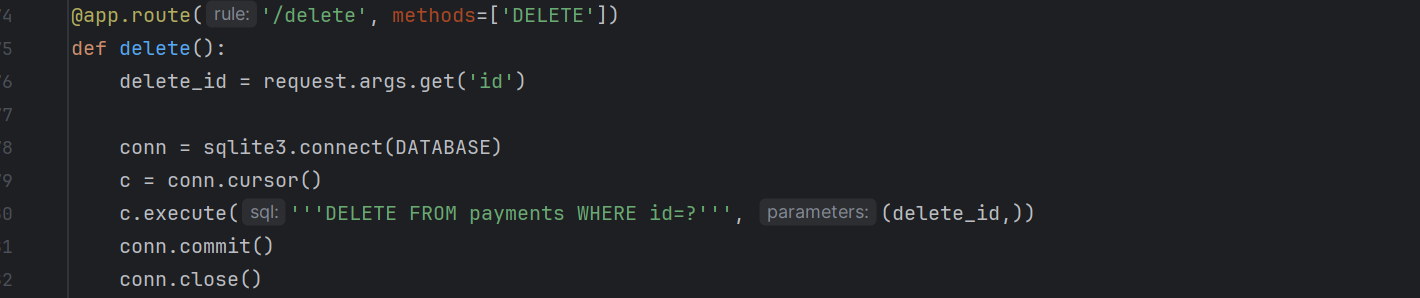
****

****

****

****

****

****

**API’s:**

1. **POST '/submit':**

Description: Submits payment data via a form submission.

Controller: submit() function.

Method: POST**.**

1. **GET '/':**

Description: Renders the main page of the application, displaying payment records and a form for submitting new payments.

Controller: index() function.

Method: GET.

1. **POST '/insert':**

Description: Inserts a new payment record via an AJAX request.

Controller: insert\_record() function.

Method: POST.

1. **GET '/summary':**

Description: Fetches payment summary data for a specific due date.

Controller: summary() function.

Method: GET.

1. **POST '/update':**

Description: Updates an existing payment record.

Controller: update() function.

Method: POST.

1. **DELETE '/delete':**

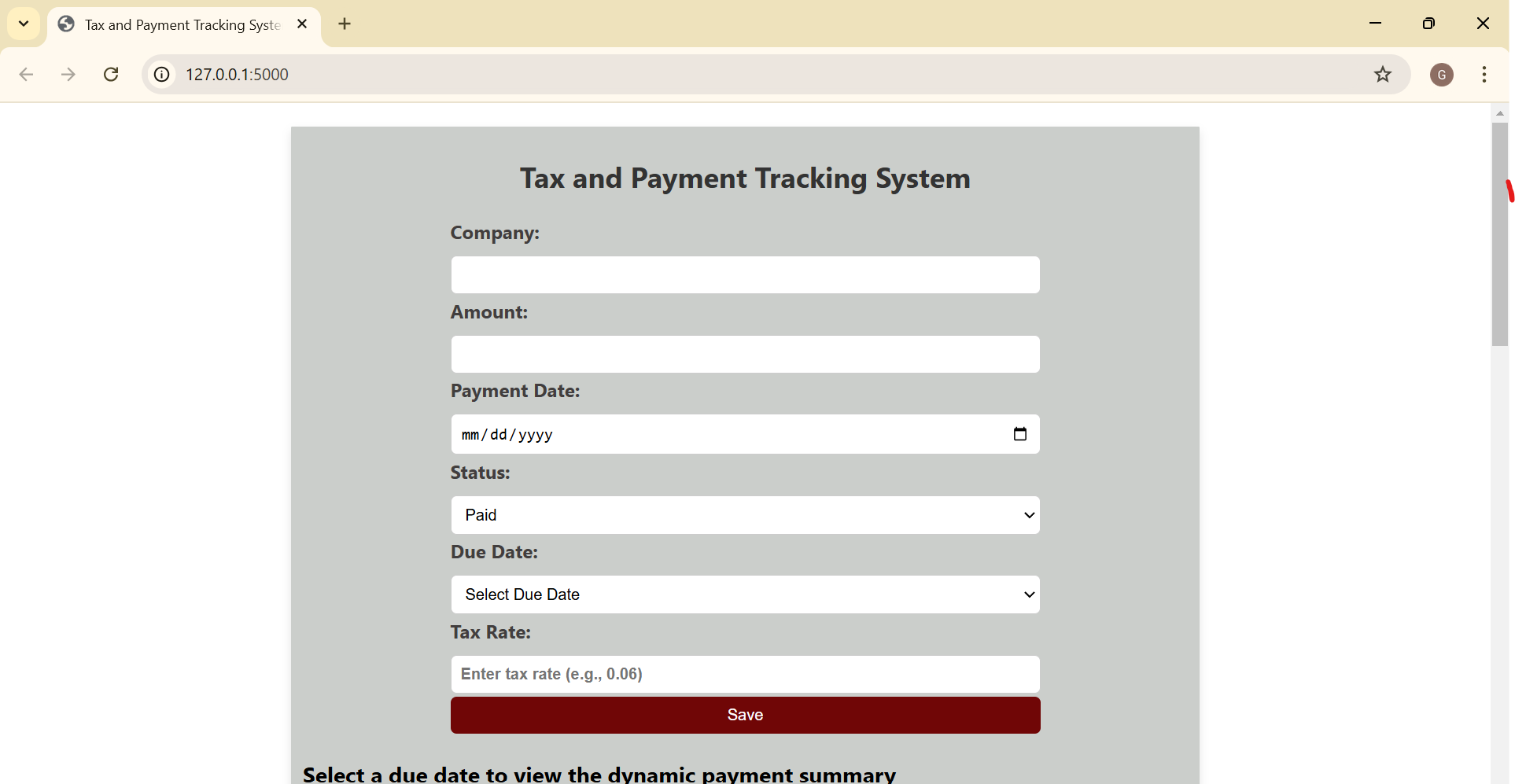
Description: Deletes an existing payment record.

Controller: delete() function.

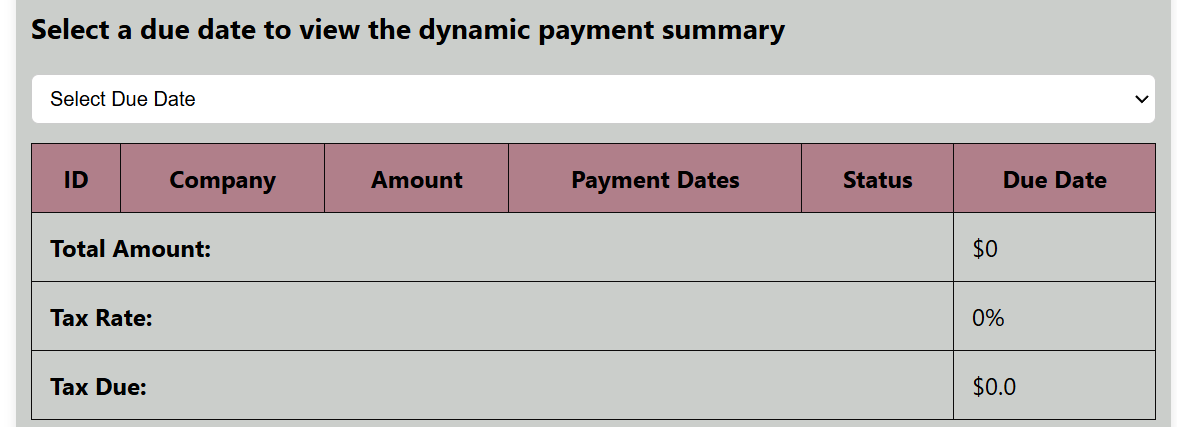
Method: DELETE.

**SCREENSHOTS**

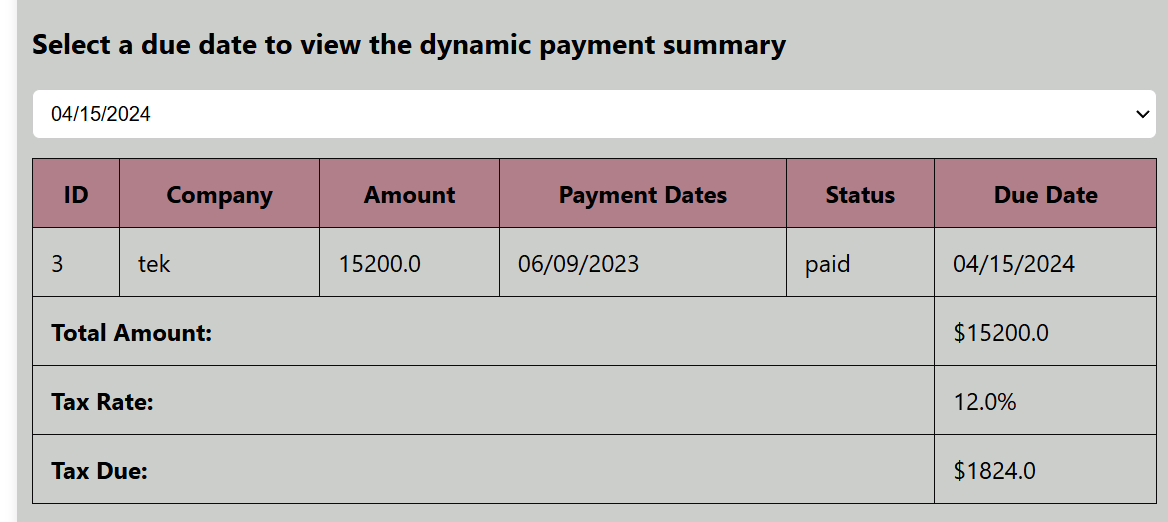
**Form to enter the data and save:**

****

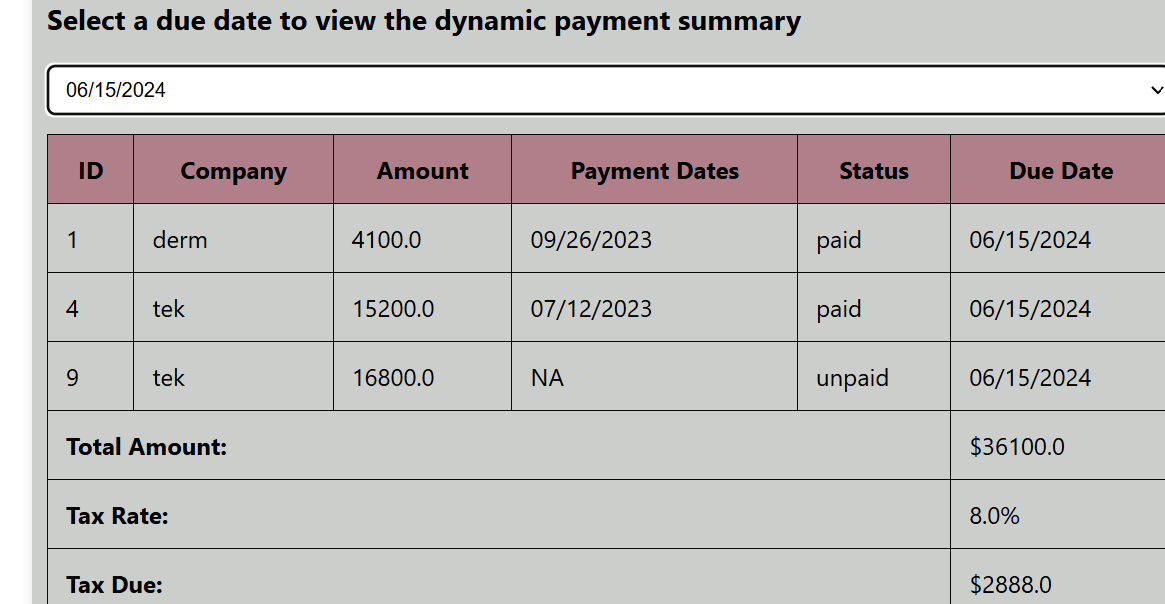
**Dynamic Payment Summary:**

****

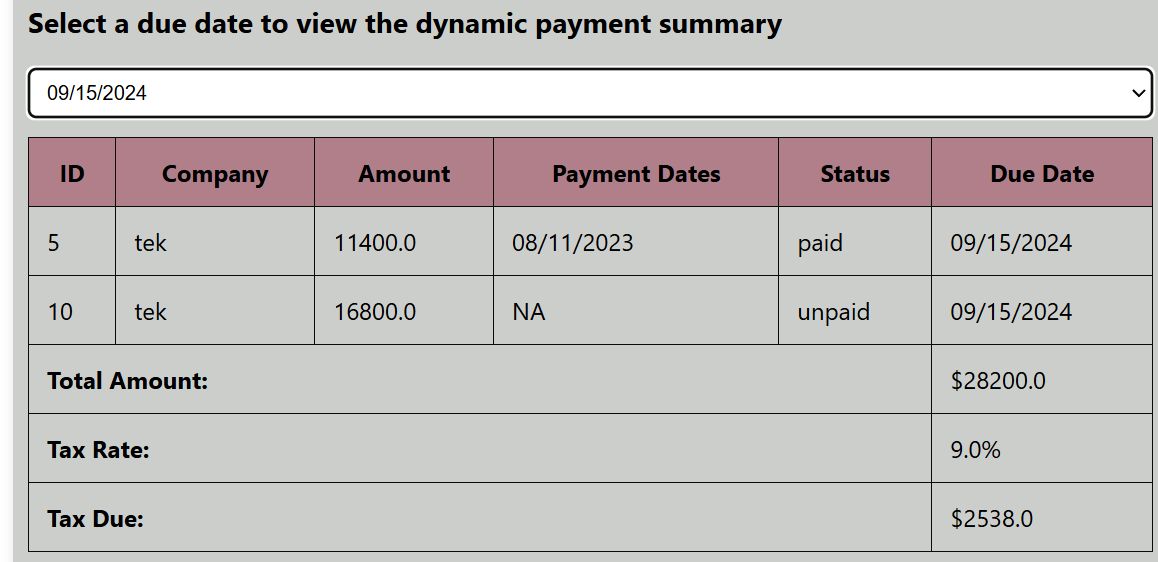
**Due Date 04/15/2024:**

****

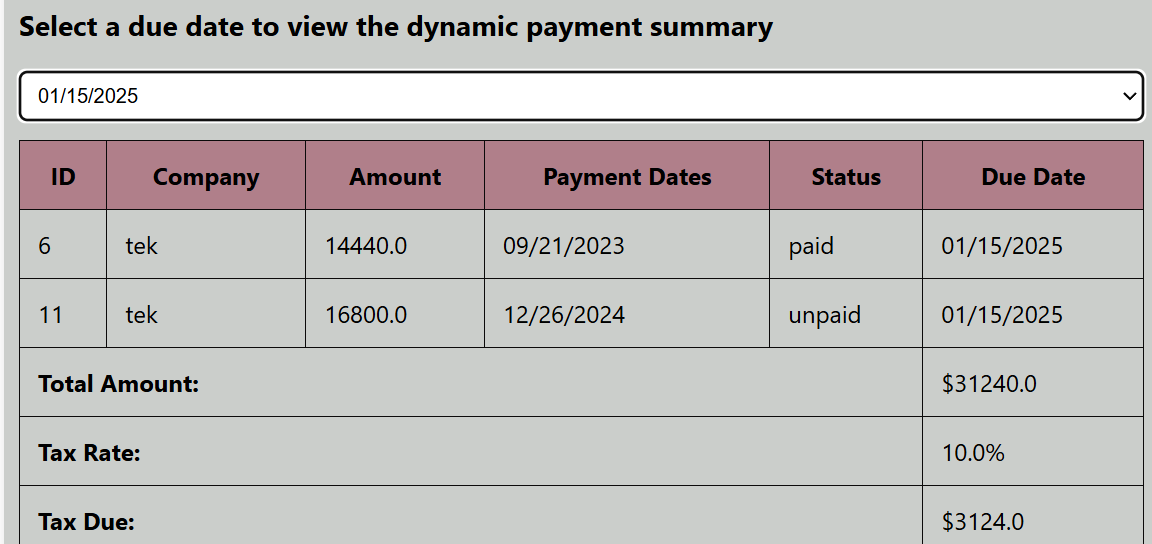
**Due Date 06/15/2024:**

****

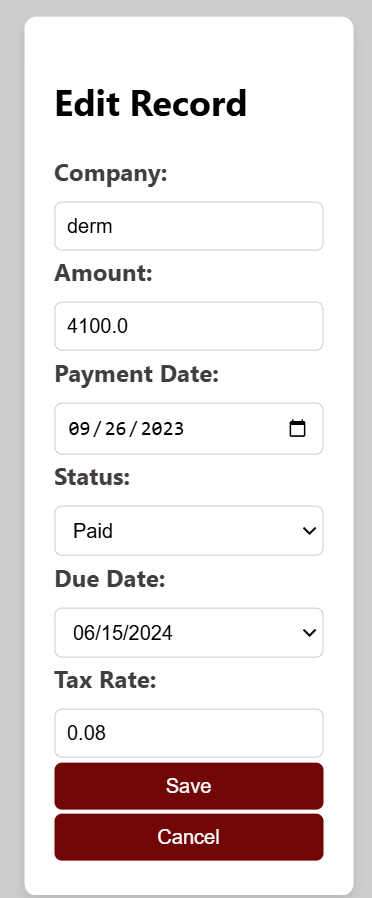
**Due Date 09/15/2024:**

****

**Due Date : 1/15/2025**

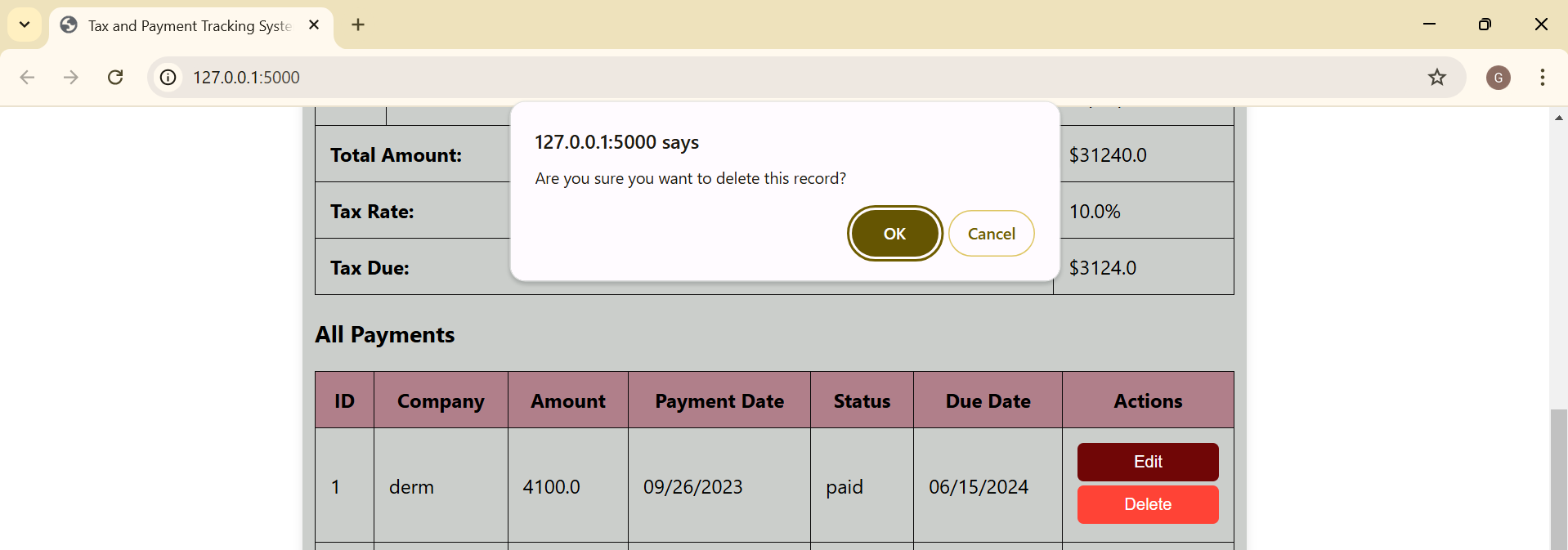
****

**Edit the data:**

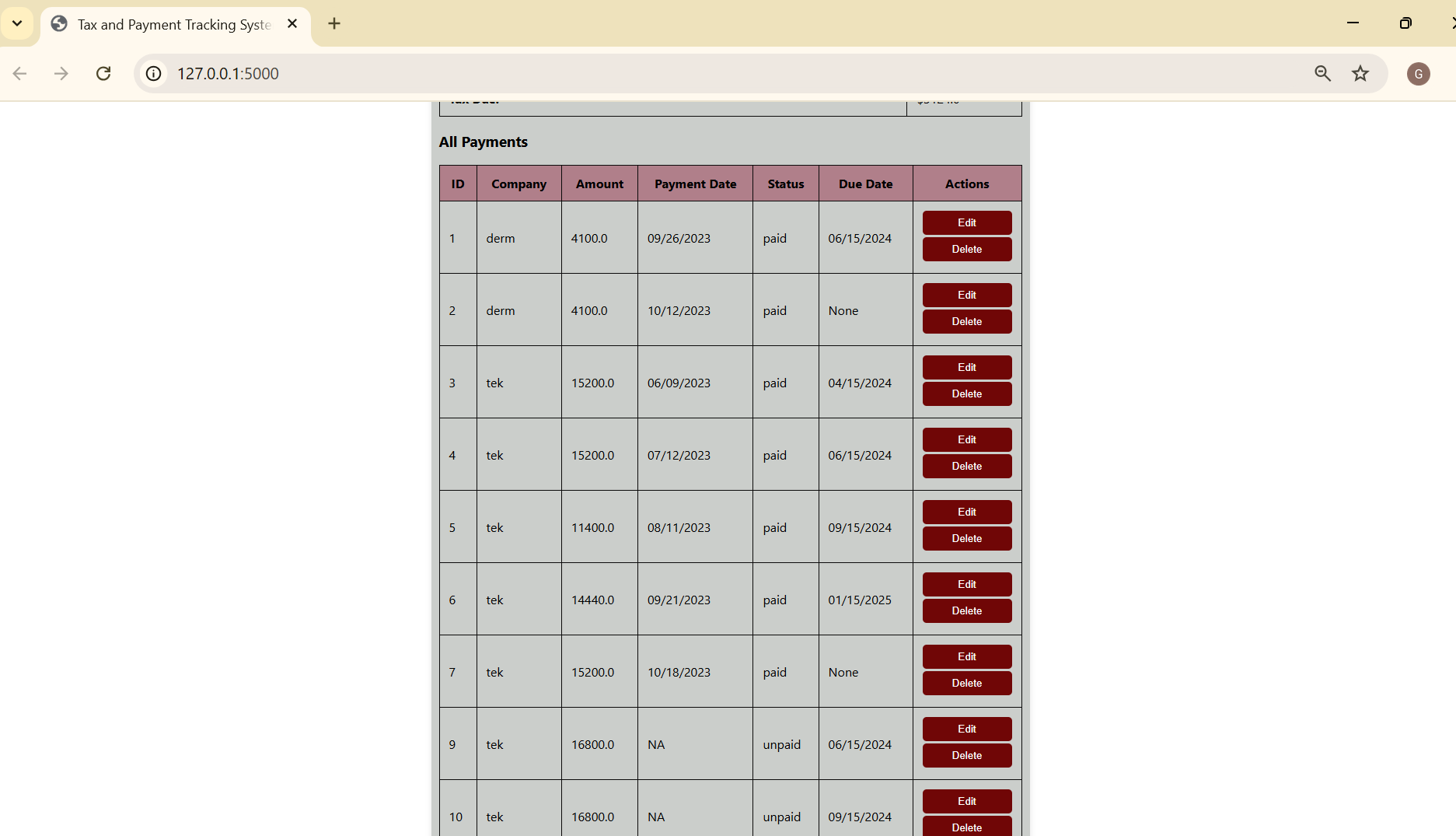
****

**Delete the data:**

**After Clicking on the delete button a pop up will a appear to confirm the operation:**

****

**ALL Payments:**

****

**GIT Repository Link:** [**https://github.com/GauriMSU/CSIT555DBMS.git**](https://github.com/GauriMSU/CSIT555DBMS.git)

**Youtube Video Link :** [**https://youtu.be/eko1Ygd8D2g**](https://youtu.be/eko1Ygd8D2g)