PES UNIVERSITY
Department of Computer Science and Engineering
UE22CS341A: Software Engineering

Deliverable 2

Implementation Document for Sales Analytics Dashboard

Prepared by
Priyanka Kumari - PES1UG22CS454
Prayashi Verma - PES1UG22CS446

PES University RR Campus

1) Creating the Database

The `init_db` function sets up a SQLite database named `sales.db` by connecting to it and creating two tables: `employees` and `sales`. The `employees` table stores details about employees, including an auto-incrementing `id`, `username`, and `role`. The `sales` table records sales transactions and includes fields for `employee_id`, customer details (name and phone), product information, purchase date and time, place, and amount. The `employee_id` in the `sales` table references the `id` in the `employees` table, ensuring that each sale is associated with a specific employee. After defining the tables, the function commits the changes to the database and closes the connection. Finally, the function is called to execute the database initialization.

```
def init_db():
    conn = sqlite3.connect('sales.db')
    cursor = conn.cursor()
    cursor.execute('''CREATE TABLE IF NOT EXISTS employees (
                        id INTEGER PRIMARY KEY AUTOINCREMENT,
                        username TEXT,
                        role TEXT
                    )''')


    cursor.execute('''CREATE TABLE IF NOT EXISTS sales (
                        id INTEGER PRIMARY KEY AUTOINCREMENT,
                        employee_id INTEGER,
                        customer_name TEXT,
                        customer_phone TEXT,
                        product TEXT,
                        purchase_date DATE,
                        purchase_time TEXT,
                        place TEXT,
                        amount REAL,
                        FOREIGN KEY (employee_id) REFERENCES employees (id)
                    )''')

    conn.commit()
    conn.close()

init_db()
```

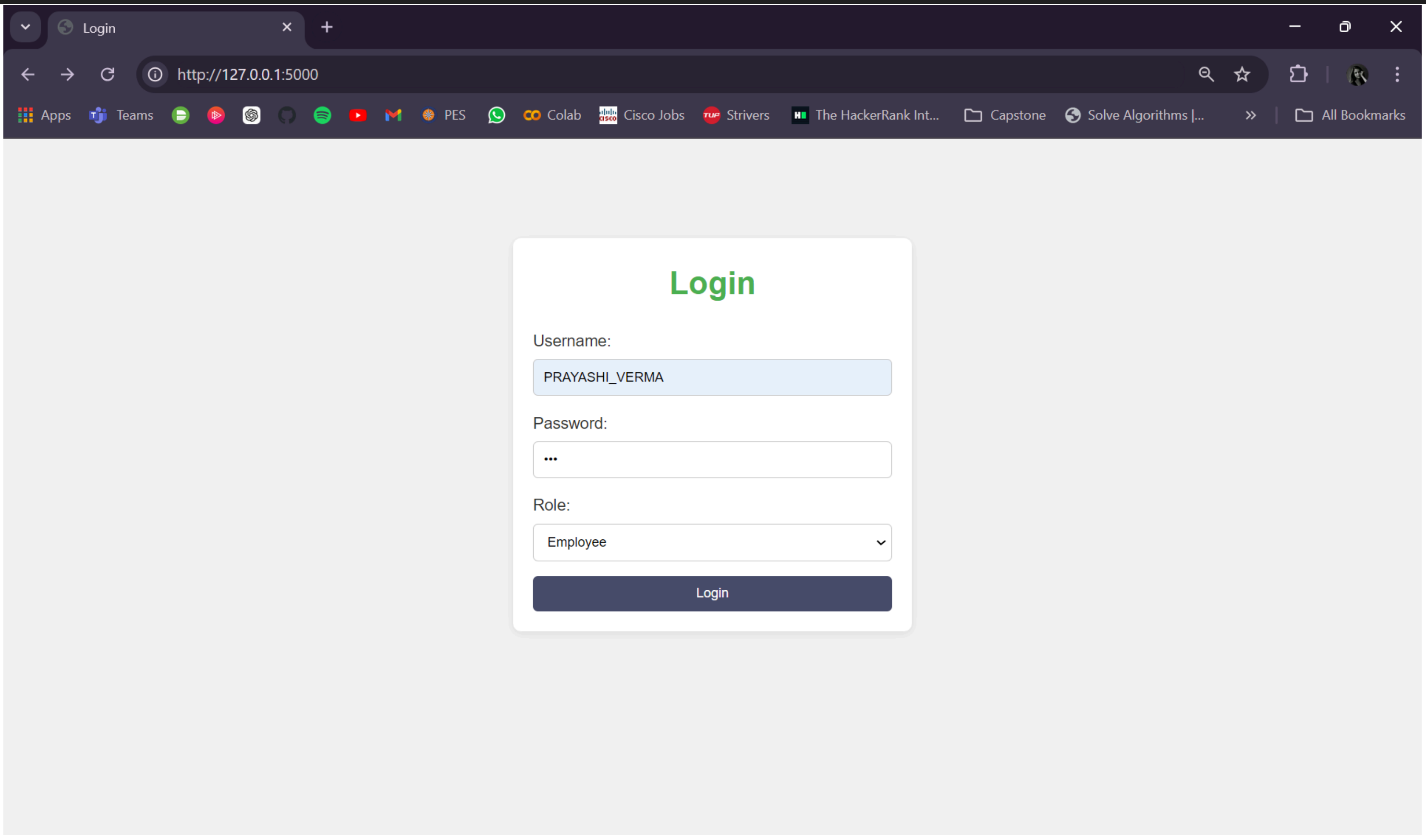
2) Creating a login page for the employee and the admin which is linked to the database

The `login` function handles user login requests in a web application. When a POST request is received, it retrieves the submitted `username` and `role` from the form. It connects to the database and checks if the username exists in the `employees` table. If the username is not found and the role is 'employee', it inserts a new employee record into the table. Afterward, it fetches the employee data again to confirm successful insertion. The connection to the database is closed once the operations are complete. If an employee is found, their ID is stored in the session, and the user is redirected to the appropriate page based on their role: either the employee dashboard, the admin page, or an informational message indicating that the analyst page is not yet implemented. If the login attempt is unsuccessful, the login form is rendered for the user to try again. The HTML page includes a simple form where users can input their username, password, and select their role before submitting the login request.

```
def login():
    if request.method == 'POST':
        username = request.form['username']
        role = request.form['role']

        conn = sqlite3.connect('sales.db')
        cursor = conn.cursor()

        cursor.execute("SELECT * FROM employees WHERE username = ?", (username,))
        employee = cursor.fetchone()

        if not employee and role == 'employee':
            cursor.execute("INSERT INTO employees (username, role) VALUES (?, ?)", (username,
role))
            conn.commit()
            cursor.execute("SELECT * FROM employees WHERE username = ?", (username,))
            employee = cursor.fetchone()

        conn.close()
        if employee:
            session['user'] = employee[0]
            if role == 'employee':
                return redirect(url_for('employee', employee_id=employee[0]))
            elif role == 'admin':
                return redirect(url_for('admin'))
            elif role == 'analyst':
                return "Analyst page not yet implemented"

    return render_template('login.html')
```

- HTML page → login.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Login</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='login_style.css') }}">
</head>
<body>
    <div class="container">
        <h1>Login</h1>
        <form method="POST">
            <label for="username">Username:</label>
            <input type="text" id="username" name="username" required>

            <label for="password">Password:</label>
            <input type="password" id="password" name="password" required>

            <label for="role">Role:</label>
            <select id="role" name="role" required>
                <option value="employee">Employee</option>
                <option value="admin">Admin</option>
                <option value="analyst">Analyst</option>
            </select>

            <input type="submit" value="Login">
        </form>
    </div>
</body>
</html>
```



Login

Username: PRAYASHI_VERMA

Password: •••

Role: Employee

Login

3) Creating Employee page and storing it in the database

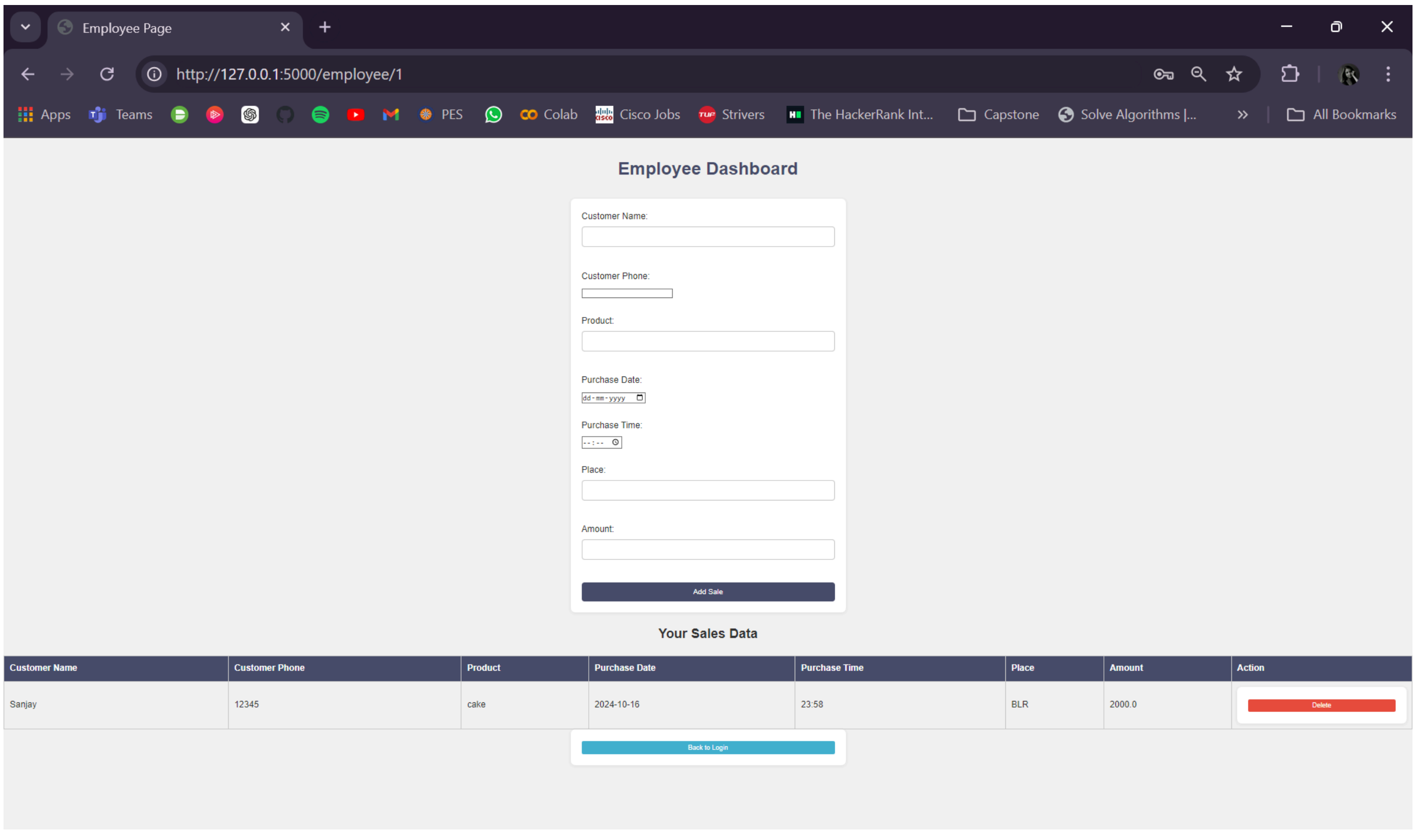
The `employee` function is a Flask route that manages the employee dashboard, allowing employees to record sales and view their sales data. When a GET or POST request is made to this route, the function first establishes a connection to the database. If the request method is POST, it retrieves customer information (name, phone), product details, purchase date and time, place of purchase, and the sale amount from the submitted form. It then inserts this data into the `sales` table, associating it with the provided `employee_id`, and commits the transaction to the database.

After handling any new sales entries, the function queries the `sales` table to retrieve all sales data related to the specific employee and closes the database connection. The retrieved sales data is then passed to the `employee.html` template for rendering. The HTML page includes a form for adding new sales, a table displaying existing sales data with options to delete specific entries, and a button to navigate back to the login page. The layout is structured to ensure a user-friendly experience for employees managing their sales records.

```
@app.route('/employee/<int:employee_id>', methods=['GET', 'POST'])
def employee(employee_id):
    conn = sqlite3.connect('sales.db')
    cursor = conn.cursor()
    if request.method == 'POST':
        customer_name = request.form['customer_name']
        customer_phone = request.form['customer_phone']
        product = request.form['product']
        purchase_date = request.form['purchase_date']
        purchase_time = request.form['purchase_time']
        place = request.form['place']
        amount = float(request.form['amount'])

        cursor.execute("INSERT INTO sales (employee_id, customer_name, customer_phone, product,
   purchase_date, purchase_time, place, amount) VALUES (?, ?, ?, ?, ?, ?, ?, ?) ",
                       (employee_id, customer_name, customer_phone, product, purchase_date,
   purchase_time, place, amount))
        conn.commit()
    cursor.execute("SELECT * FROM sales WHERE employee_id = ?", (employee_id,))
    sales_data = cursor.fetchall()
    conn.close()
    return render_template('employee.html', sales_data=sales_data, employee_id=employee_id)
```

- HTML page →employee.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Employee Page</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='employee_style.css') }}">
</head>
<body>
    <h1>Employee Dashboard</h1>
    <form method="POST">
        <label for="customer_name">Customer Name:</label>
        <input type="text" id="customer_name" name="customer_name" required><br><br>

        <label for="customer_phone">Customer Phone:</label>
        <input type="tel" id="customer_phone" name="customer_phone" required><br><br>

        <label for="product">Product:</label>
        <input type="text" id="product" name="product" required><br><br>

        <label for="purchase_date">Purchase Date:</label>
        <input type="date" id="purchase_date" name="purchase_date" required><br><br>

        <label for="purchase_time">Purchase Time:</label>
        <input type="time" id="purchase_time" name="purchase_time" required><br><br>

        <label for="place">Place:</label>
        <input type="text" id="place" name="place" required><br><br>

        <label for="amount">Amount:</label>
        <input type="number" step="0.01" id="amount" name="amount" required><br><br>

        <input type="submit" value="Add Sale">
    </form>

    <h2>Your Sales Data</h2>
    <table>
        <tr>
            <th>Customer Name</th>
            <th>Customer Phone</th>
            <th>Product</th>
            <th>Purchase Date</th>
            <th>Purchase Time</th>
            <th>Place</th>
            <th>Amount</th>
            <th>Action</th>
        </tr>
        {% for sale in sales_data %}
        <tr>
            <td>{{ sale[2] }}</td> <!-- Customer Name -->
            <td>{{ sale[3] }}</td> <!-- Customer Phone -->
            <td>{{ sale[4] }}</td> <!-- Product -->
            <td>{{ sale[5][:10] }}</td> <!-- Purchase Date -->
            <td>{{ sale[6] }}</td> <!-- Purchase Time -->
            <td>{{ sale[7] }}</td> <!-- Place -->
            <td>{{ sale[8] }}</td> <!-- Amount -->
            <td>
                <!-- Form to delete the sale -->
                <form action="{{ url_for('delete_sale', sale_id=sale[0],
   employee_id=employee_id) }}" method="POST">
                    <input type="submit" value="Delete" style="font-size: 12px; padding: 5px
   10px; background-color: #e74c3c; color: white; border: none; border-radius: 3px; cursor:
   pointer;">
                </form>
            </td>
        </tr>
        {% endfor %}
    </table>

    <form action="{{ url_for('login') }}">
        <input type="submit" value="Back to Login" style="font-size: 12px; padding: 5px 10px;
   background-color: #44b2d1; color: white; border: none; border-radius: 3px; cursor:
   pointer;">
    </form>
</body>
</html>
```



Your Sales Data

Customer Name	Customer Phone	Product	Purchase Date	Purchase Time	Place	Amount	Action
Sanjay	12345	cake	2024-10-16	23:58	BLR	2000.0	Delete

4) Given an option for the employee to delete the customer detail that they have added.

The `delete_sale` function is a Flask route that handles the deletion of a specific sale record from the `sales` database. When a POST request is received, it connects to the database and creates a cursor for executing SQL commands. The function takes two parameters: `sale_id`, which identifies the specific sale to be deleted, and `employee_id`, which links back to the employee managing the sale.

It executes a SQL DELETE statement to remove the sale record corresponding to the provided `sale_id` and commits the changes to the database. After successfully deleting the record, the database connection is closed. The function then redirects the user back to the employee dashboard by calling the `employee` route with the specified `employee_id`, allowing them to view the updated list of sales.

```python
@app.route('/delete_sale/<int:sale_id>/<int:employee_id>', methods=['POST'])
def delete_sale(sale_id, employee_id):
    conn = sqlite3.connect('sales.db')
    cursor = conn.cursor()

    cursor.execute("DELETE FROM sales WHERE id = ?", (sale_id,))
    conn.commit()
    conn.close()

    return redirect(url_for('employee', employee_id=employee_id))
```

5) Creating an admin login who can view all the employee data i.e. what are the contents that they have added in the database

The `admin` function is a Flask route designed to display an admin dashboard that provides an overview of employees and their sales data. When a GET request is made to this route, the function connects to the database and creates a cursor for executing SQL queries. It performs a SQL query that retrieves all employees along with their corresponding sales records using a LEFT JOIN on the `employees` and `sales` tables, ensuring that all employees are included even if they have no sales.

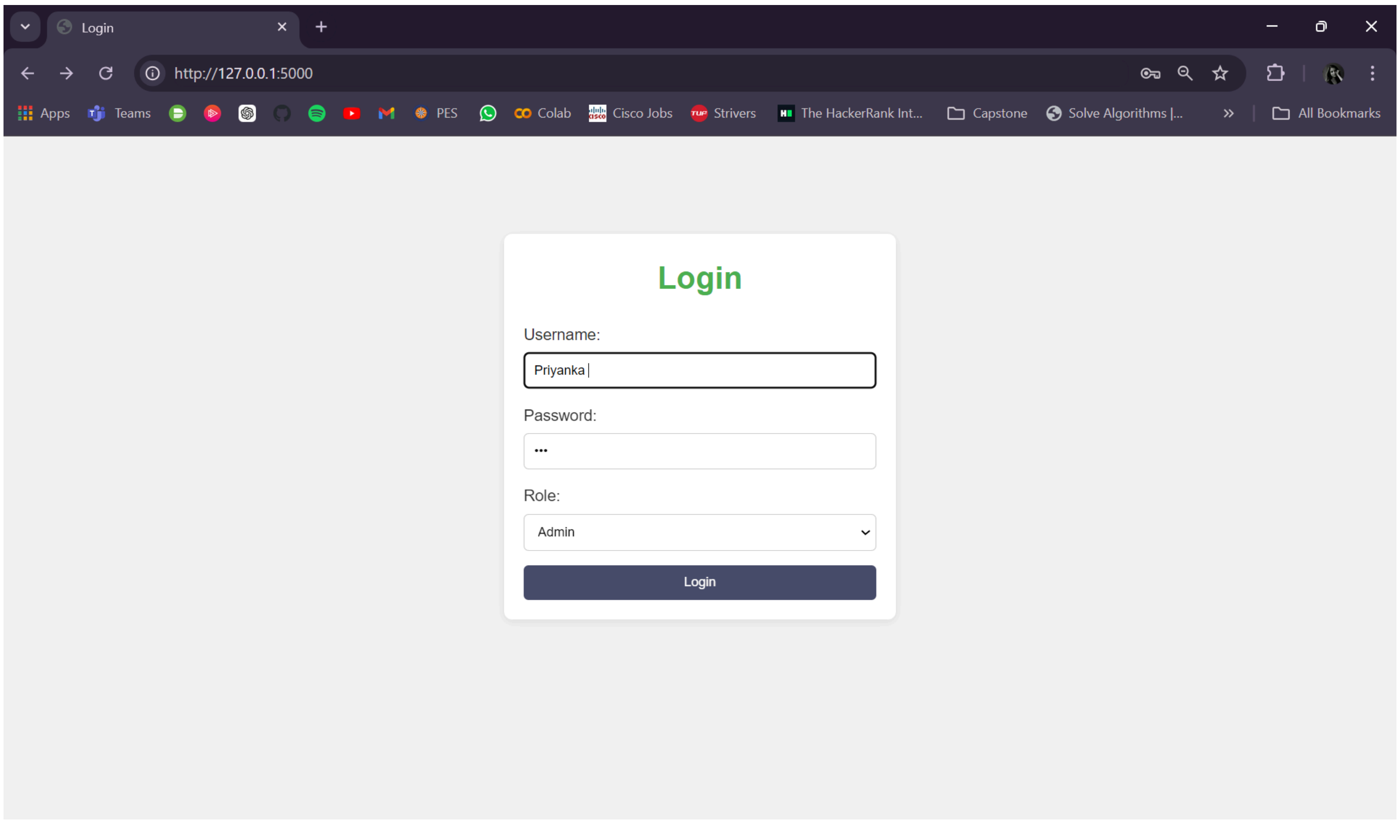
The fetched data is stored in the `employee_sales_data` variable, which is then processed to organize the information into a dictionary called `employees`. Each employee's ID is used as a key, and their username and associated sales are stored as values. If an employee has sales data, it is appended to their respective list.

After processing the data, the database connection is closed. Finally, the function renders the `admin.html` template, passing the `employees` dictionary to display each employee's name and their sales information in a structured format. The HTML page features an organized layout with a table for each employee, showing relevant details for each sale, and includes a button to navigate back to the login page.

```
@app.route('/admin', methods=['GET'])
def admin():
    conn = sqlite3.connect('sales.db')
    cursor = conn.cursor()


    cursor.execute('''
        SELECT employees.id, employees.username, sales.customer_name, sales.customer_phone,
   sales.product, sales.purchase_date, sales.purchase_time, sales.place, sales.amount
        FROM employees
        LEFT JOIN sales ON employees.id = sales.employee_id
    ''')
    employee_sales_data = cursor.fetchall()

    conn.close()
    employees = {}
    for data in employee_sales_data:
        employee_id = data[0]
        username = data[1]
        sale = data[2:]

        if employee_id not in employees:
            employees[employee_id] = {
                'username': username,
                'sales': []
            }
        if sale[0] is not None:
            employees[employee_id]['sales'].append(sale)

    return render_template('admin.html', employees=employees)
```

- HTML page → admin.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Admin Dashboard</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">
</head>
<body>
    <h1>Admin Dashboard</h1>
    <h2>Employees and Their Sales</h2
    {% for employee_id, employee_data in employees.items() %}
        <div class="employee-section">
            <h3>Employee: {{ employee_data['username'] }}</h3>

            <table border="1">
                <thead>
                    <tr>
                        <th>Customer Name</th>
                        <th>Customer Phone</th>
                        <th>Product</th>
                        <th>Purchase Date</th>
                        <th>Purchase Time</th>
                        <th>Place</th>
                        <th>Amount</th>
                    </tr>
                </thead>
                <tbody>
                    {% for sale in employee_data['sales'] %}
                    <tr>
                        <td>{{ sale[0] }}</td> <!-- Customer Name -->
                        <td>{{ sale[1] }}</td> <!-- Customer Phone -->
                        <td>{{ sale[2] }}</td> <!-- Product -->
                        <td>{{ sale[3][:10] }}</td> <!-- Purchase Date -->
                        <td>{{ sale[4] }}</td> <!-- Purchase Time -->
                        <td>{{ sale[5] }}</td> <!-- Place -->
                        <td>{{ sale[6] }}</td> <!-- Amount -->
                    </tr>
                    {% endfor %}
                </tbody>
            </table>
        </div>
    {% endfor %}

    <form action="{{ url_for('login') }}">
        <input type="submit" value="Back to Login" style="font-size: 12px; padding: 5px 10px;
   background-color: #44b2d1; color: white; border: none; border-radius: 3px; cursor:
   pointer;">
    </form>
</body>
</html>
```



Login

http://127.0.0.1:5000

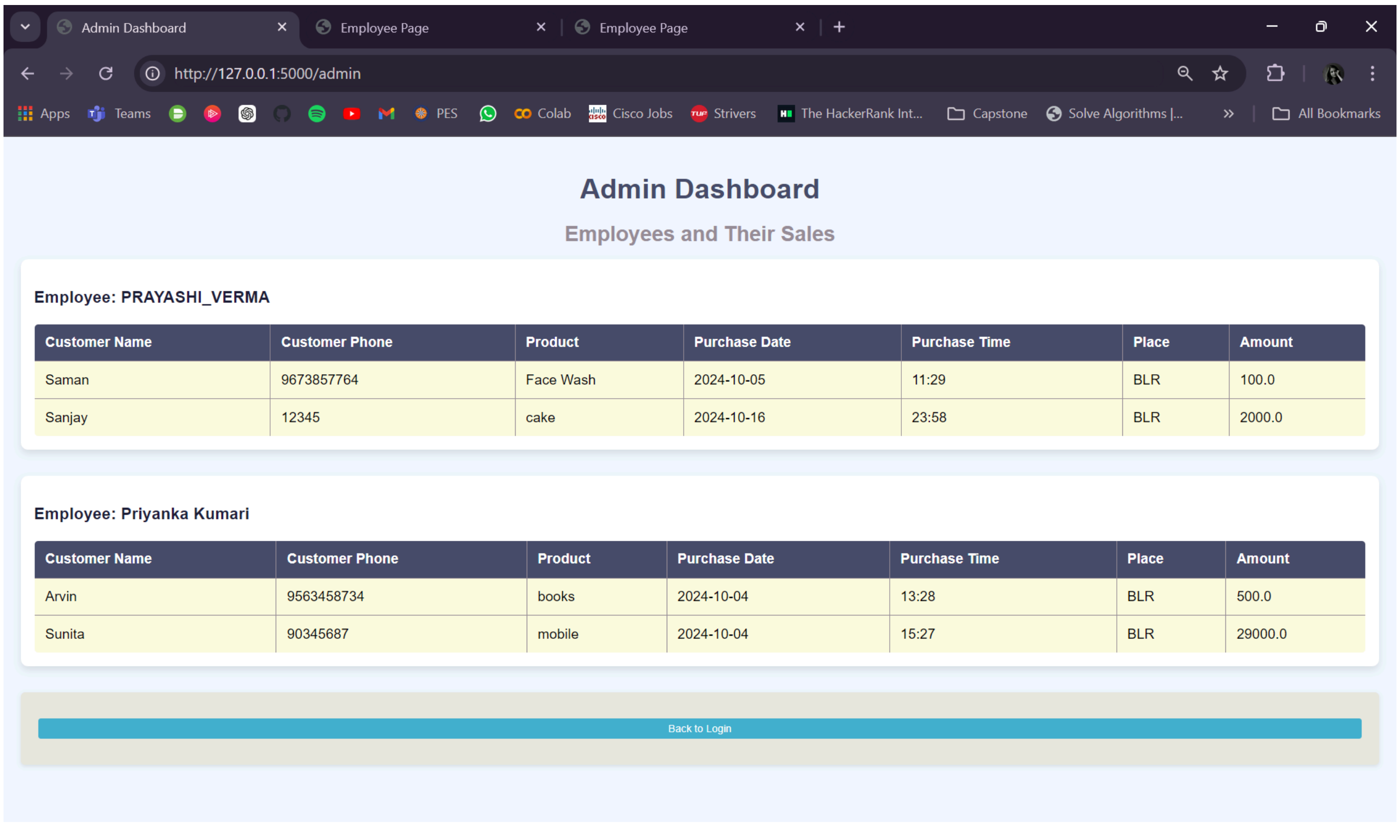
Login

Username: Priyanka

Password: •••

Role: Admin

Login



Admin Dashboard

http://127.0.0.1:5000/admin

Admin Dashboard

Employees and Their Sales

Employee: PRAYASHI_VERMA

Customer Name	Customer Phone	Product	Purchase Date	Purchase Time	Place	Amount
Saman	9673857764	Face Wash	2024-10-05	11:29	BLR	100.0
Sanjay	12345	cake	2024-10-16	23:58	BLR	2000.0

Employee: Priyanka Kumari

Customer Name	Customer Phone	Product	Purchase Date	Purchase Time	Place	Amount
Arvin	9563458734	books	2024-10-04	13:28	BLR	500.0
Sunita	90345687	mobile	2024-10-04	15:27	BLR	29000.0

Back to Login