TSE2101
Individual Report

for

Quotation System — Finance Officer

Version 1.0

Tutorial Section: TT2L

Group No: 4

Srihari Naidu A/L Venkatash **1201100182**

Date: 27/11/23

CONTENTS

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
Draft Type and Number	Full Name	Information about the revision. This table does not need to be filled in whenever a document is touched, only when the version is being upgraded.	00/00/00

1 System Overview

1.1 Description

The role of a Finance Officer in this system is a simple and straightforward dealing with Purchase Order(s).The major process of our system's Finance Officer, is keying-in or submitting Purchase Order(s) in the system.

1.2 Use Cases

View Purchase Order
Submit Purchase Order

1.3 Assumptions and Dependencies

Assumptions

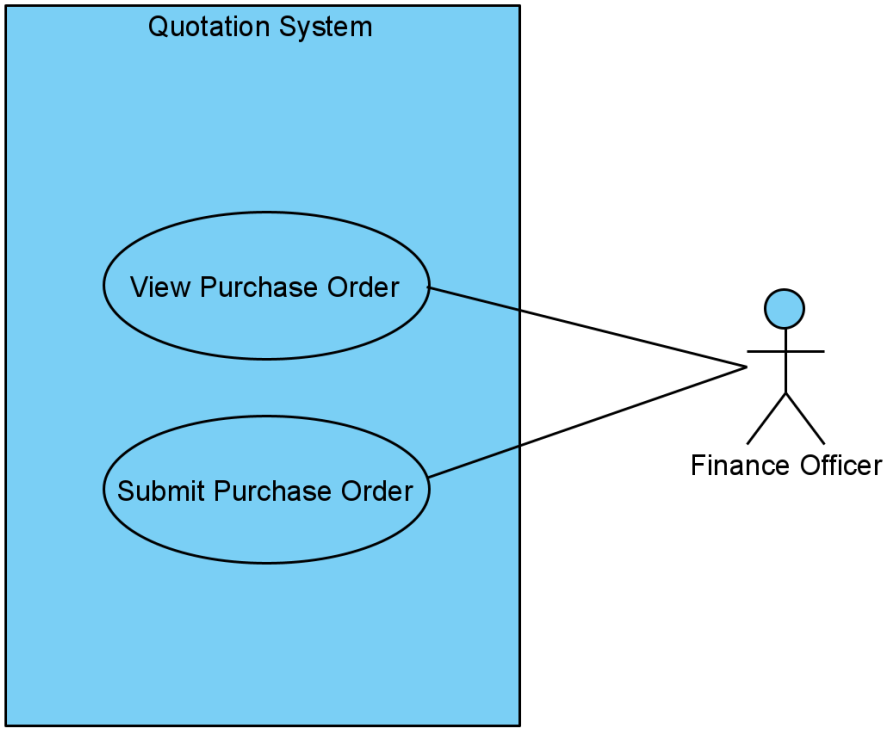
- The Finance Officer is registered into the system by an administrator. Hence, won't/can't register an account by ownself.
- The Finance Officer receives a "*physical/real-life copy"* of the Purchase Order by the Customer, which is to be keyed-in.

Dependencies

- The Finance Officer must already have an account created by the administrator, to use the system.

2 Requirements

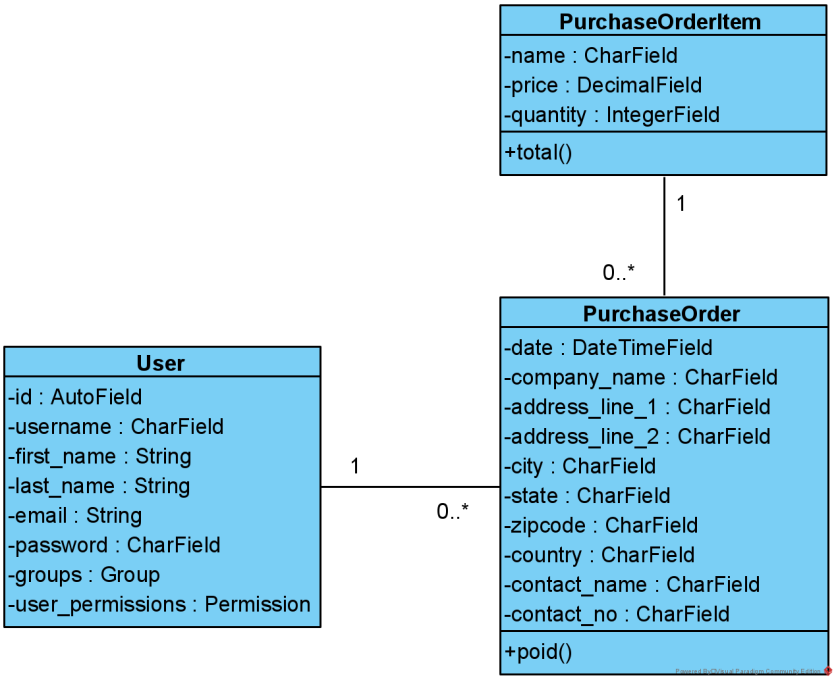
2.1 Use Case Diagram



Use Case Diagram - Finance Officer

The Use Case Diagram in the scope of the system's Finance Officer shows 2 main/major use cases that the Finance Officer has.

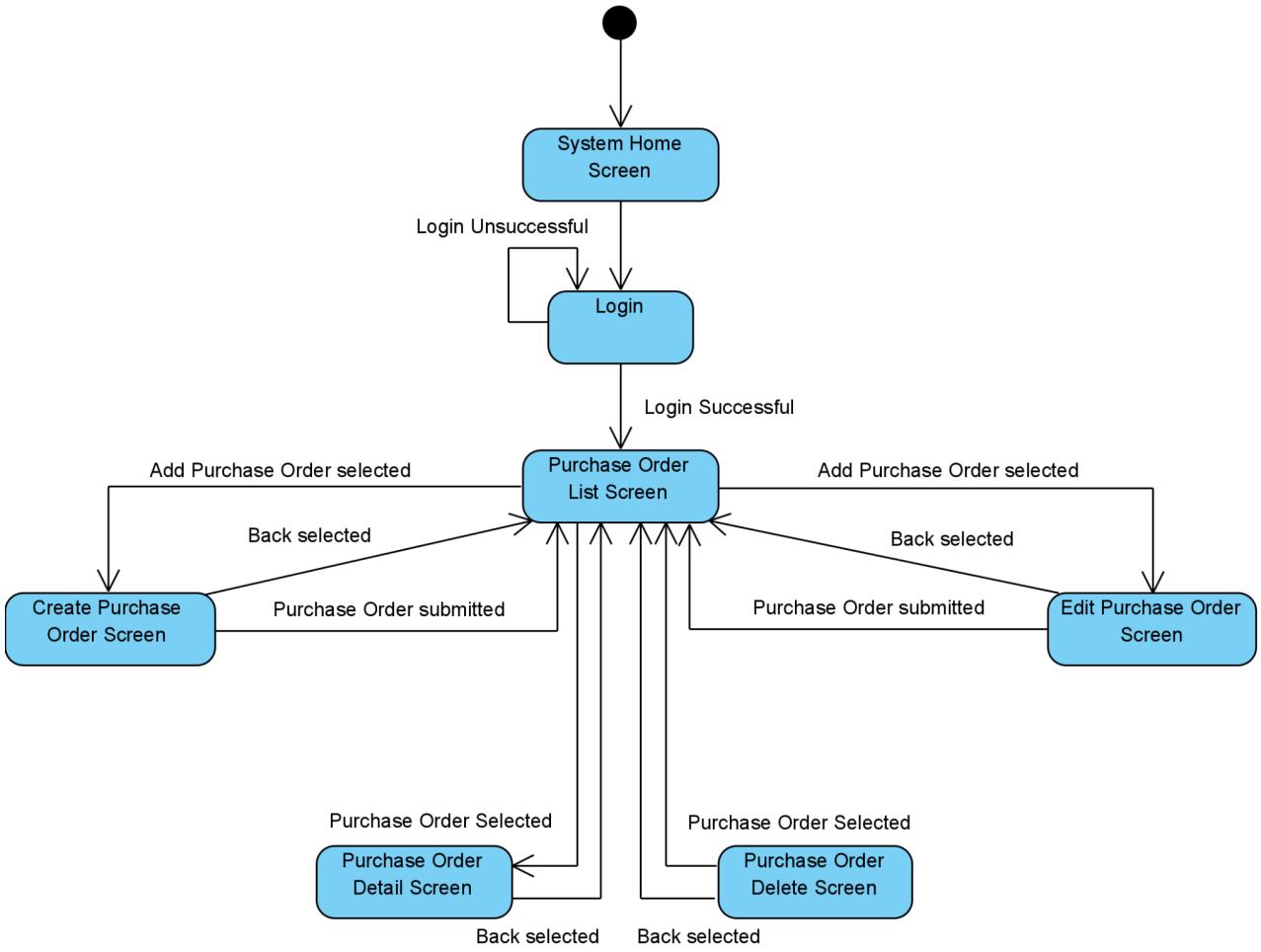
2.2 Class Diagram



Class Diagram - Finance Officer

2.3 State Diagrams

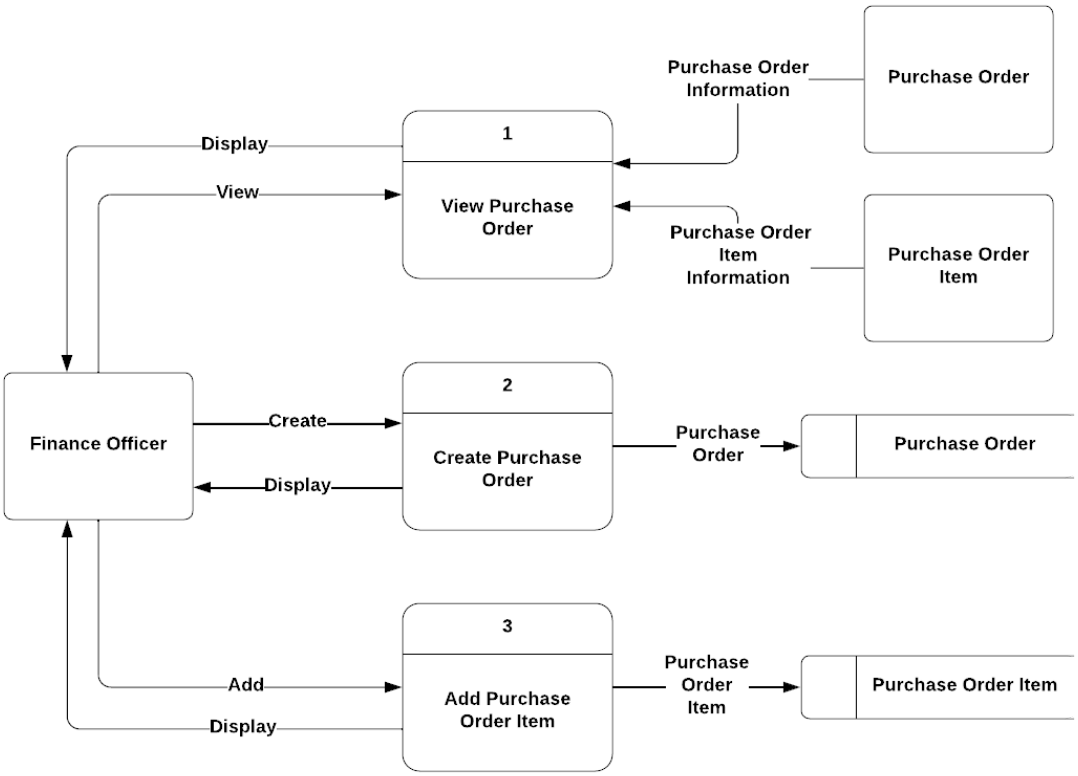
<TO DO: Describe the states, events, and transitions relevant to the use cases and place the state diagrams here.>



State Diagram - Finance Officer

2.4 Data Flow Diagrams

<TO DO: Describe the data flows relevant to the use cases and place the data flow diagrams here.>

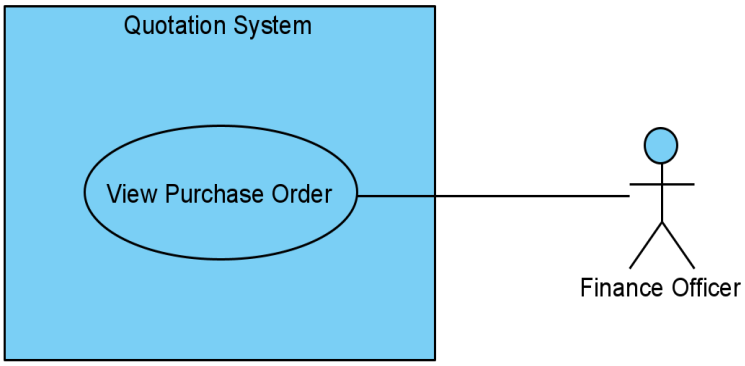


Data Flow Diagram - Finance Officer

3 Design

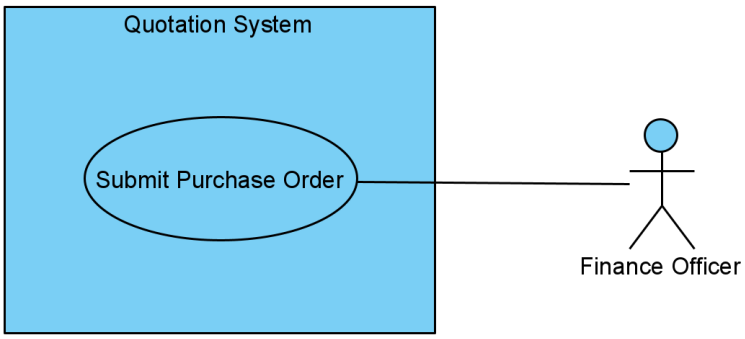
3.1 Use Cases

3.1.1 View Purchase Order



Use Case: View Purchase Order
Brief description: View Purchase Order sent by the Customer.
Primary actors: Finance Officer
Secondary actors: None
Preconditions: 1. The Finance Officer is logged in the system.
Main flow: 1. The Finance Officer launches the system. 2. The Finance Officer logs in the system and is navigated to the homepage. 3. The Finance Officer selects “View Purchase Order”. 4. The Finance Officer views the Purchase Order form with the Item ID, Name, Quantity and Price.
Postconditions: 1. The purchase order is viewed by the Finance Officer.
Alternative flows: The Finance Officer cancels the process. The Finance Officer closes the program.

3.1.2 Submit Purchase Order



Use Case: Submit Purchase Order
Brief description: Submit & record customer's Purchase Order in the system.
Primary actors: Finance Officer
Secondary actors: None
Preconditions: 1. Finance Officer has logged in the system.
Main flow: 1. The Finance Officer launches the system. 2. The Finance Officer logs in the system and is navigated to the homepage. 3. The Finance Officer selects "View Purchase Order". 4. The Finance Officer views the Purchase Order form with the Item ID, Name, Quantity and Price.
Postconditions: 1. The purchase order is viewed by the Finance Officer.
Alternative flows: 1. The Finance Officer cancels the process. 2. The Finance Officer closes the program.

3.2 Data Dictionary

3.3 Data Structures

3.3.1 Purchase Order

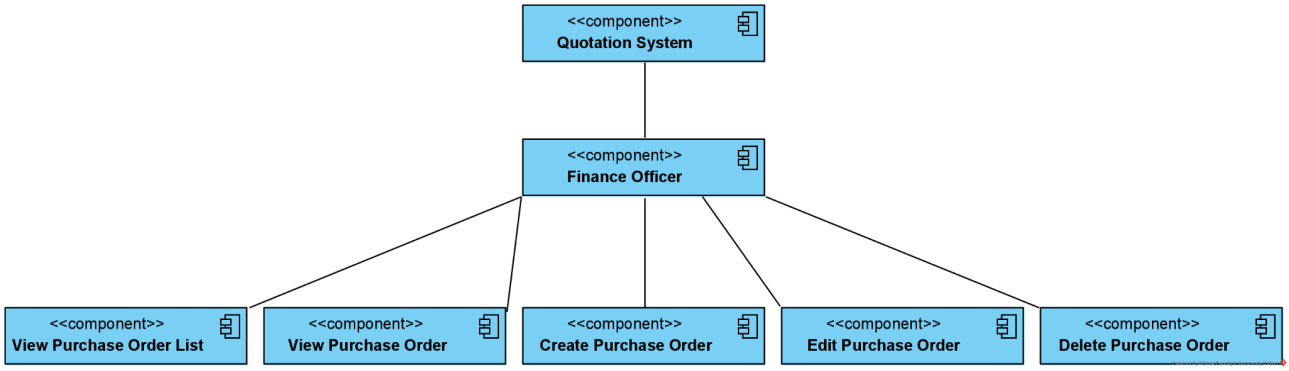
Data Structure: Purchase Order				
Attribute Name	**Description**	**Data Type**	**Required**	**Key**
id	Purchase order id	int	Y	PK
officer	Owner user	int	Y	FK
date	Purchase Order date	date	Y	
company_name	Customer's company name	Varchar(100	Y	
address_line_1	Customer's Delivery address	Varchar(100	Y	
address_line_2		Varchar(100)	Y	
city		Varchar(50)	Y	
state		Varchar(20)	Y	
zipcode		Varchar(10)	Y	
country		Varchar(20)	Y	
contact_name	Customer's Contact number	Varchar(50)	Y	
contact_no	Customer's Contact number	Varchar(20)	Y	

3.3.2 Purchase Order Item

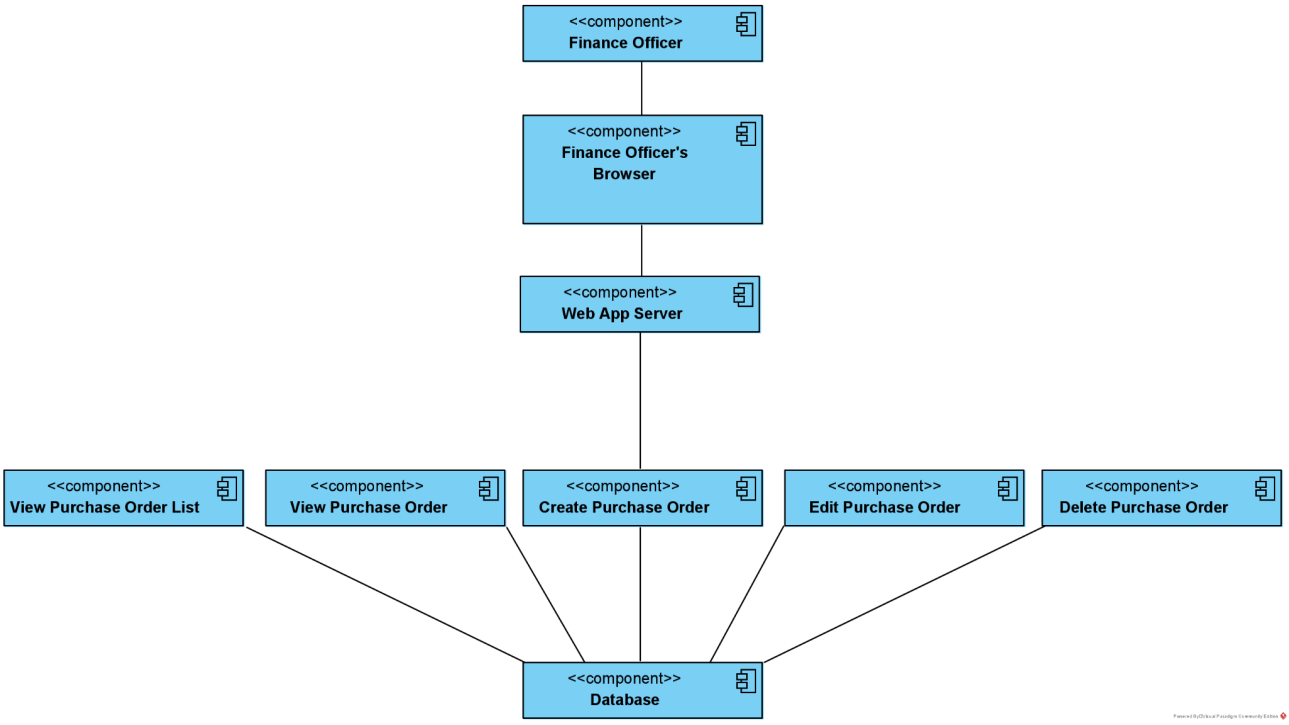
Data Structure: Purchase Order Item				
Attribute Name	**Description**	**Data Type**	**Required**	**Key**
id	Purchase Order Item ID	int	Y	PK
po	Owning Purchase Order	int	Y	FK
name	Item’s name	Varchar(100	Y	
price	item price	int	Y	
quantity	item quantity	int	Y	

3.4 Subsystem Architecture

<TO DO: Describe the subsystem and place the architecture diagram here.>



Subsystem Architecture 1 - Finance Officer

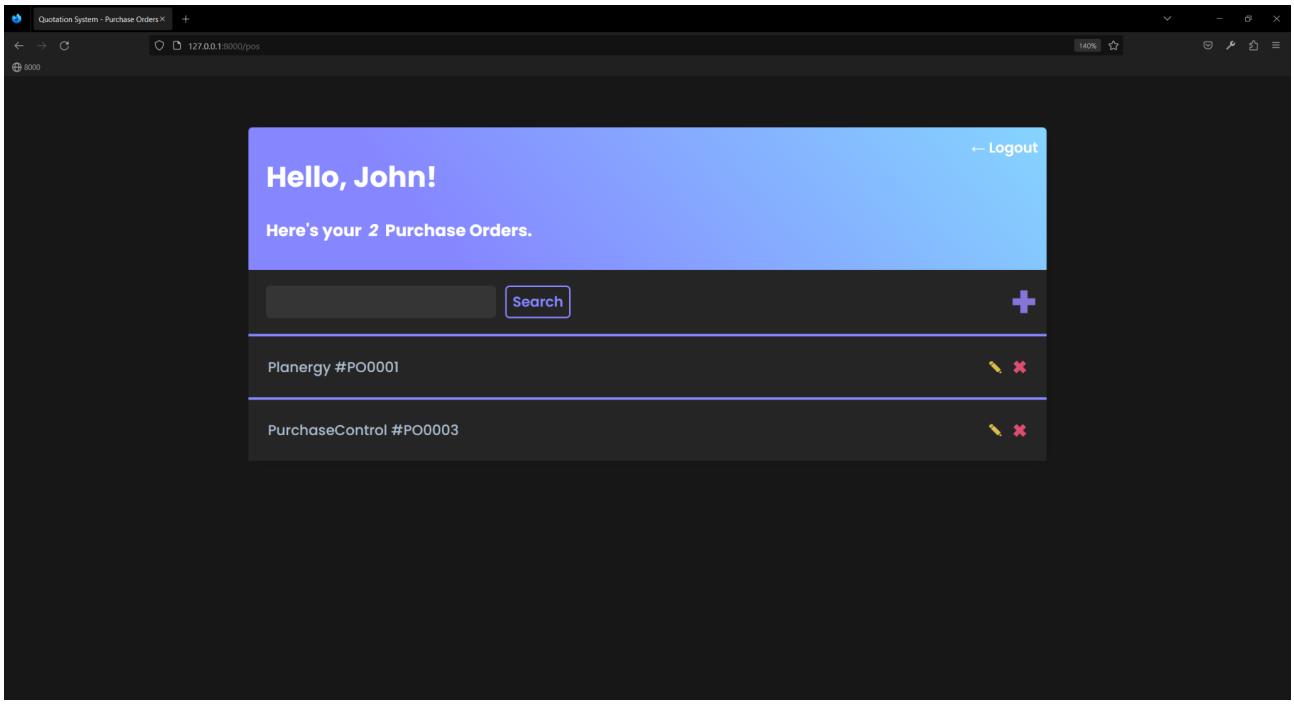


Subsystem Architecture 2 - Finance Officer

3.5 Subsystem Screens

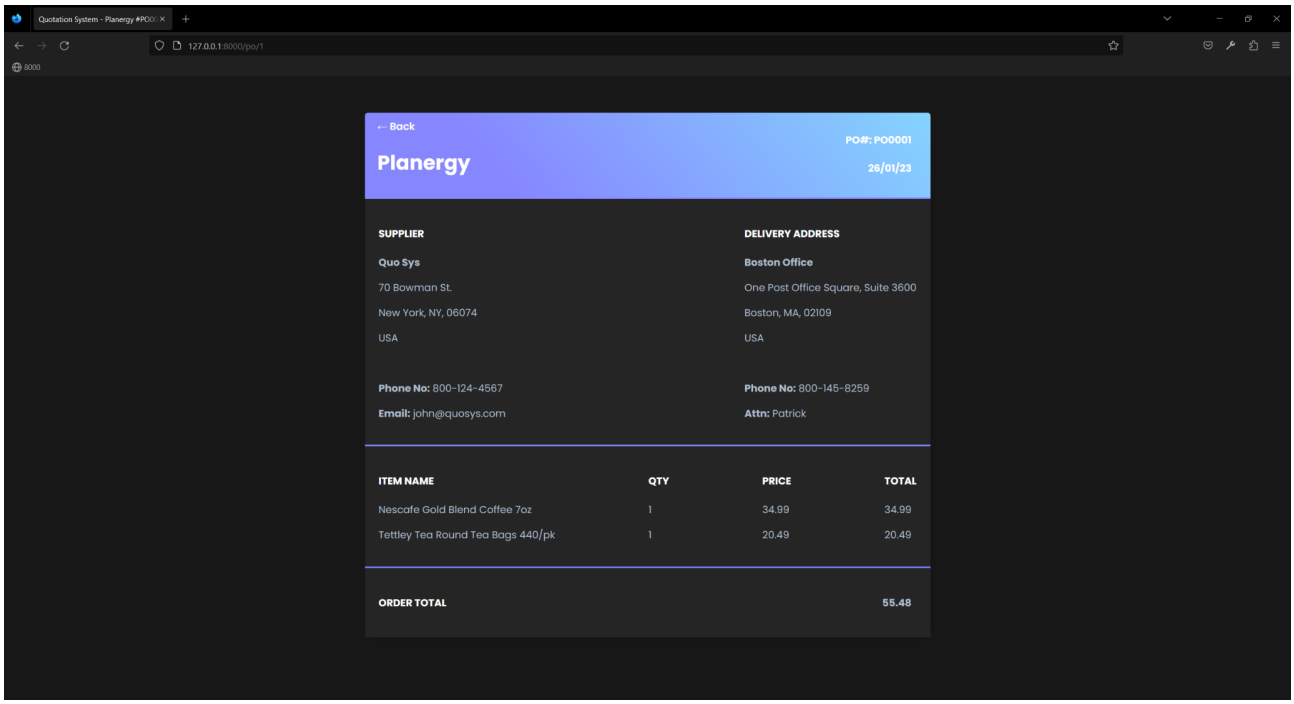
3.5.1 View Purchase Order List

This screen can be considered as the “dashboard’ for the Finance Officer where the user views their submissions of Purchase Orders.



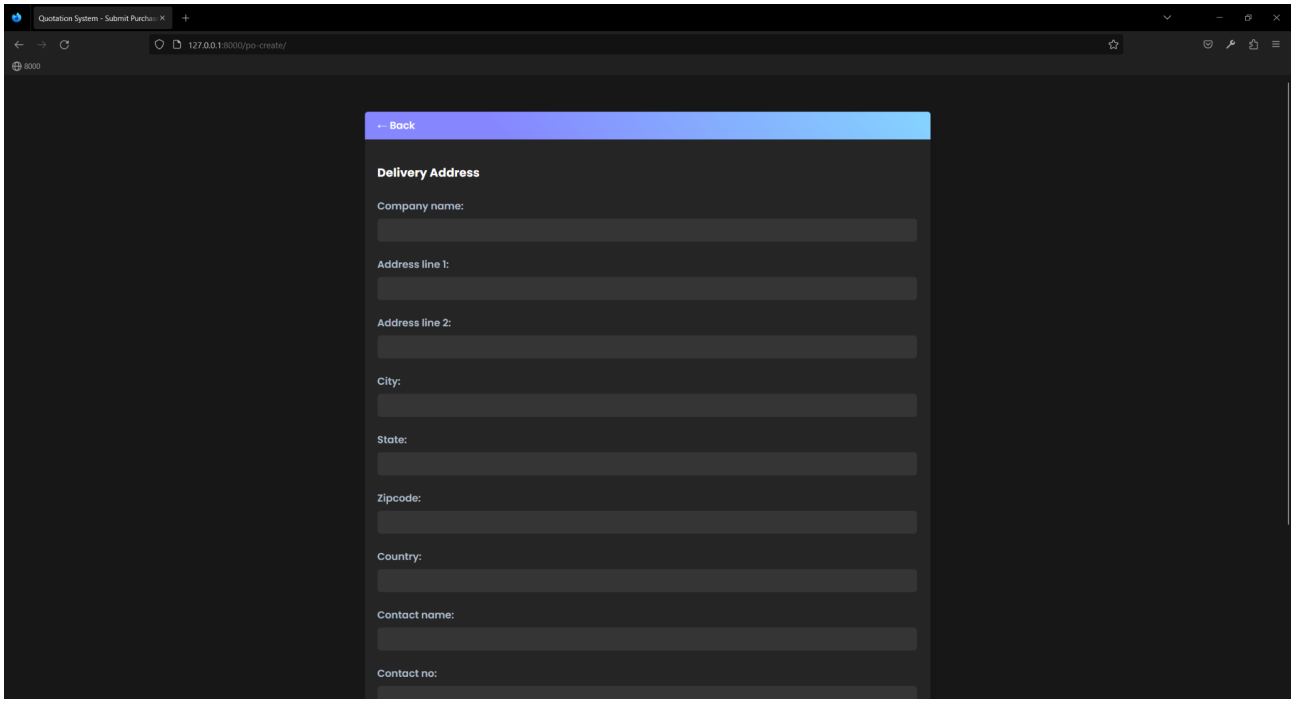
3.5.2 View Purchase Order

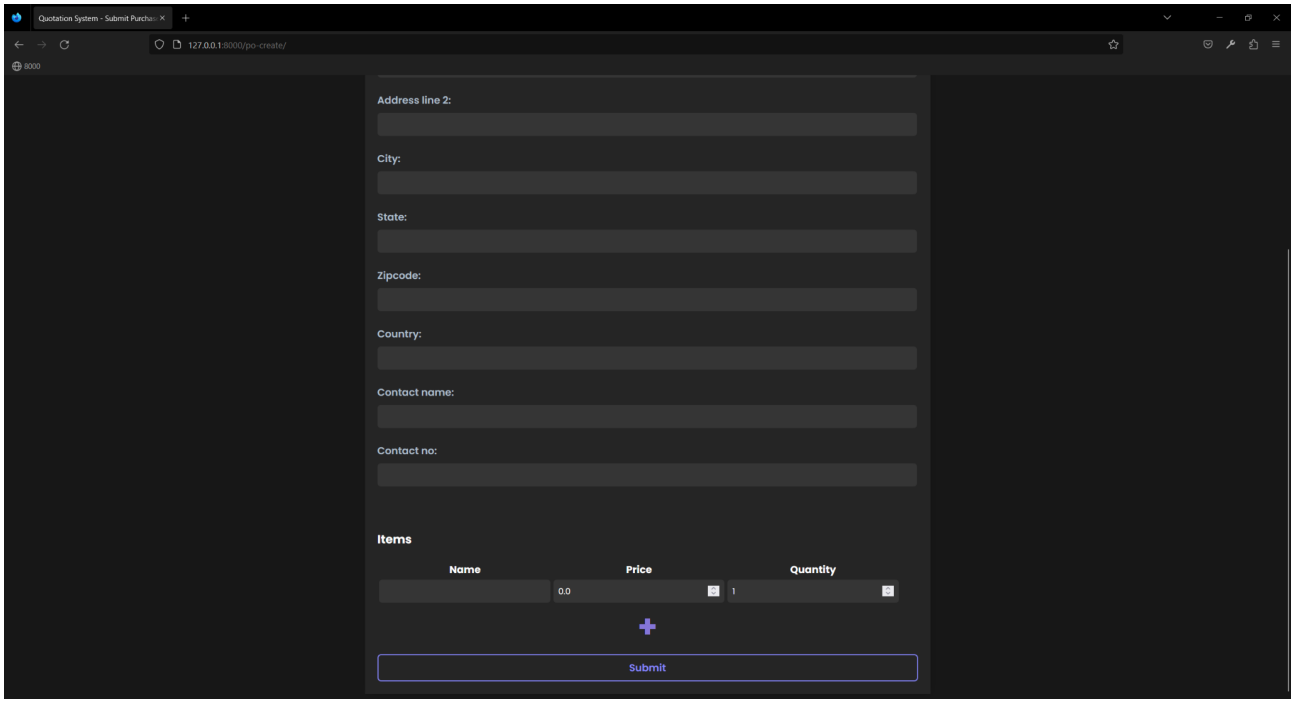
This screen is the detailed view of a Purchase Order selected by the Finance Officer.



3.5.3 Create Purchase Order

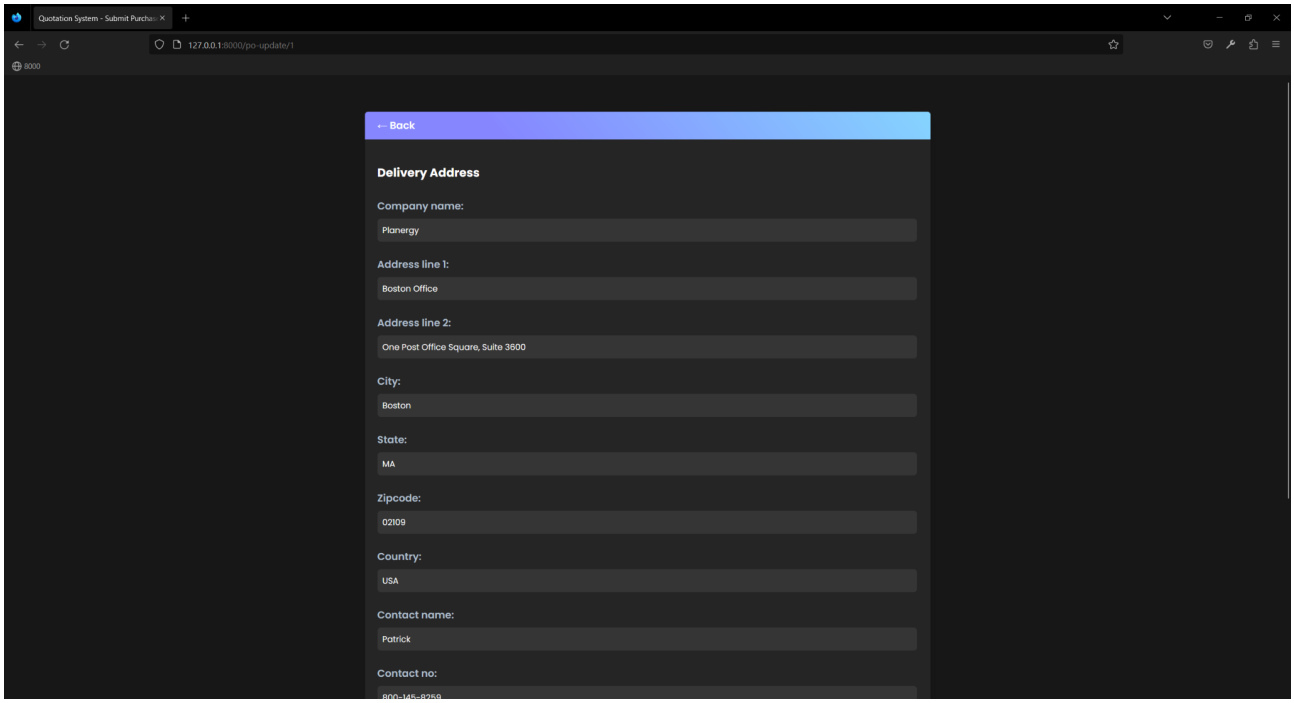
This screen is the form to create a Purchase Order with fields to fill in the delivery address and items.

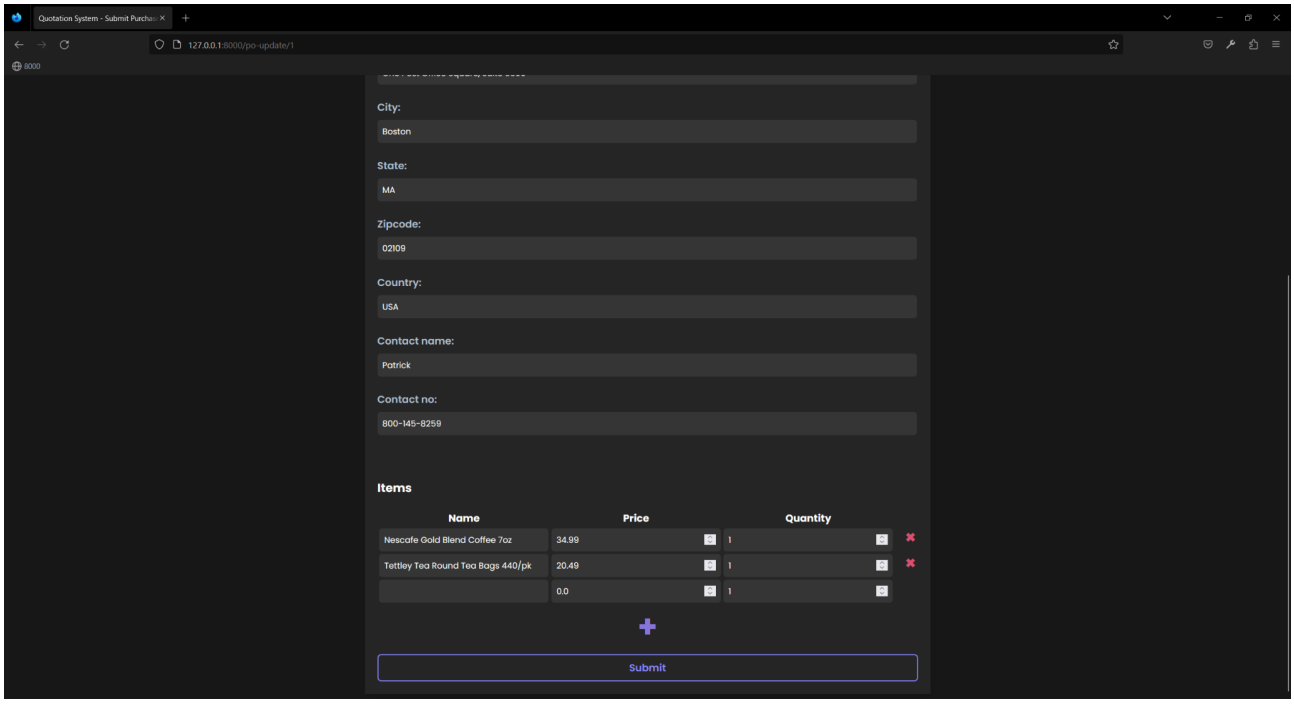




3.5.4 Edit Purchase Order

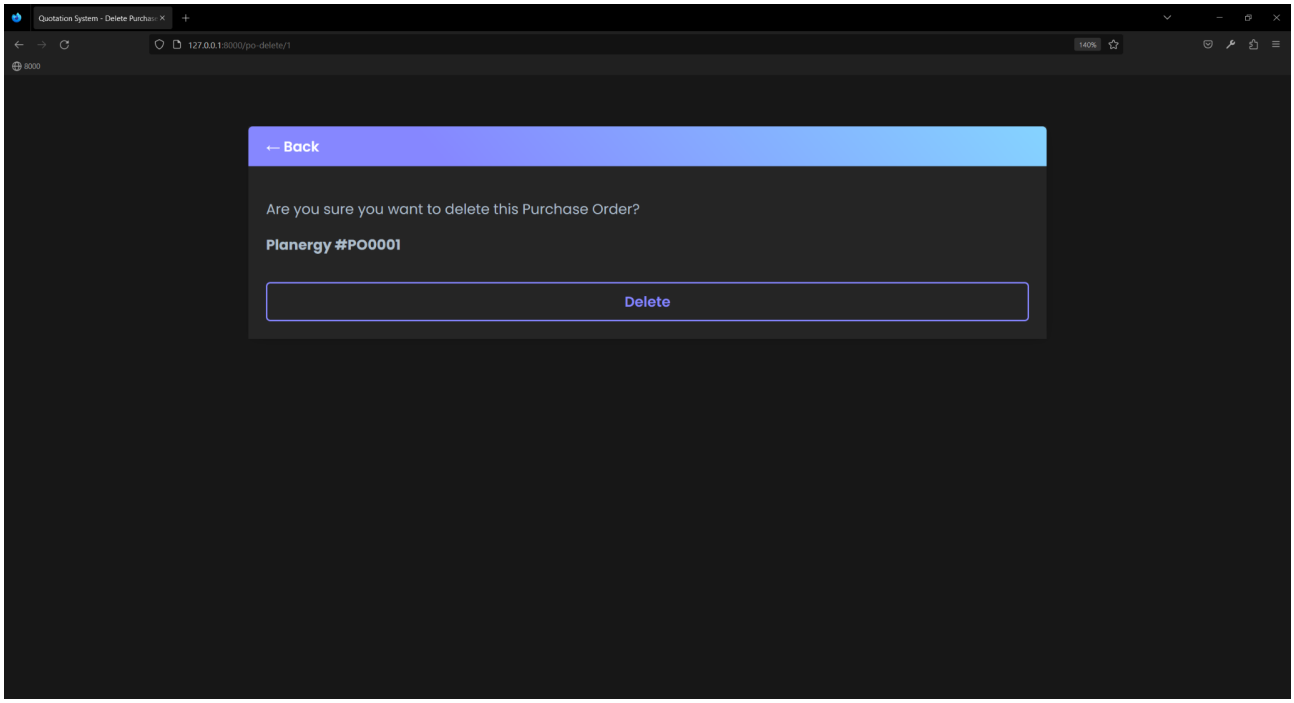
This screen is similar to the Create Purchase Order screen but the fields are pre-filled with the existing data.





3.5.5 Delete Purchase Order

This screen is more of a confirmation page where the Finance Officer chooses to proceed with deleting the selected Purchase Order.



3.6 Subsystem Components

3.6.1 View Purchase Order List

This component can be considered as the “dashboard" for the Finance Officer where the user views their submissions of Purchase Orders.

Pseudocode

1. Login.
2. Get all Purchase Orders submitted by the current user.
3. Display all Purchase Orders.
4. Provide functions to Add, Edit, Delete and View a Purchase Order.

3.6.2 View Purchase Order

This component is the detailed view of a Purchase Order selected by the Finance Officer.

Pseudocode

1. Click on a Purchase Order.
2. Get the selected Purchase Order from Database.
3. Display the Purchase Order in a formatted way.

3.6.3 Create Purchase Order

This component is to create a Purchase Order with fields to fill in the delivery address and items.

Pseudocode

1. Click on Add Purchase Order.
2. Show form to fill in Purchase Order.
3. Show form to fill in items of Purchase Order.
4. Save Purchase Order to database upon click of Submit.

3.6.4 Edit Purchase Order

This component is to edit a Purchase Order with fields to edit the delivery address and items.

Pseudocode

1. Click on Edit Purchase Order.
2. Fill in fields with existing data.
3. Save changes to database upon click of Submit.

3.6.5 Delete Purchase Order

This component is to edit a Purchase Order with fields to edit the delivery address and items.

Pseudocode

1. Click on Delete Purchase Order.
2. Get the selected Purchase Order from the database.
3. Prompt Officer confirmation to delete purchase order.
4. Delete Purchase Order from database upon confirmation.

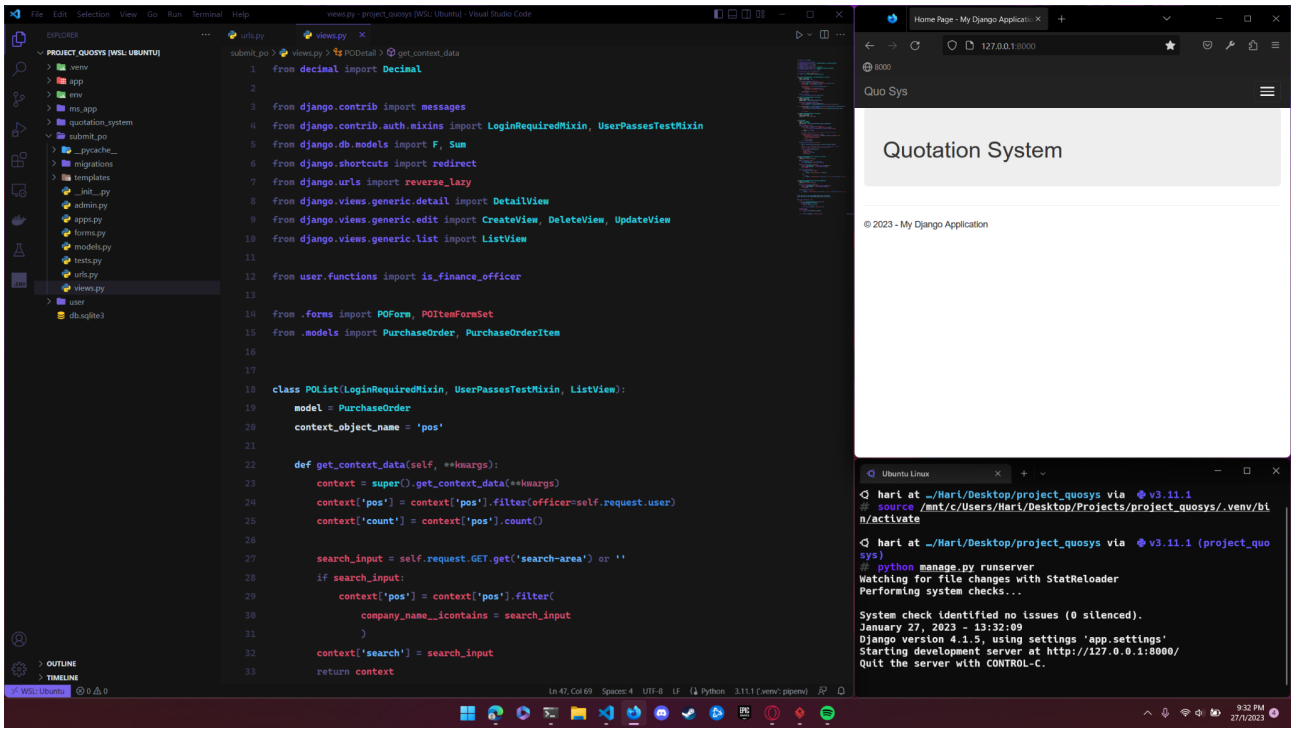
4 Implementation

4.1 Development Environment

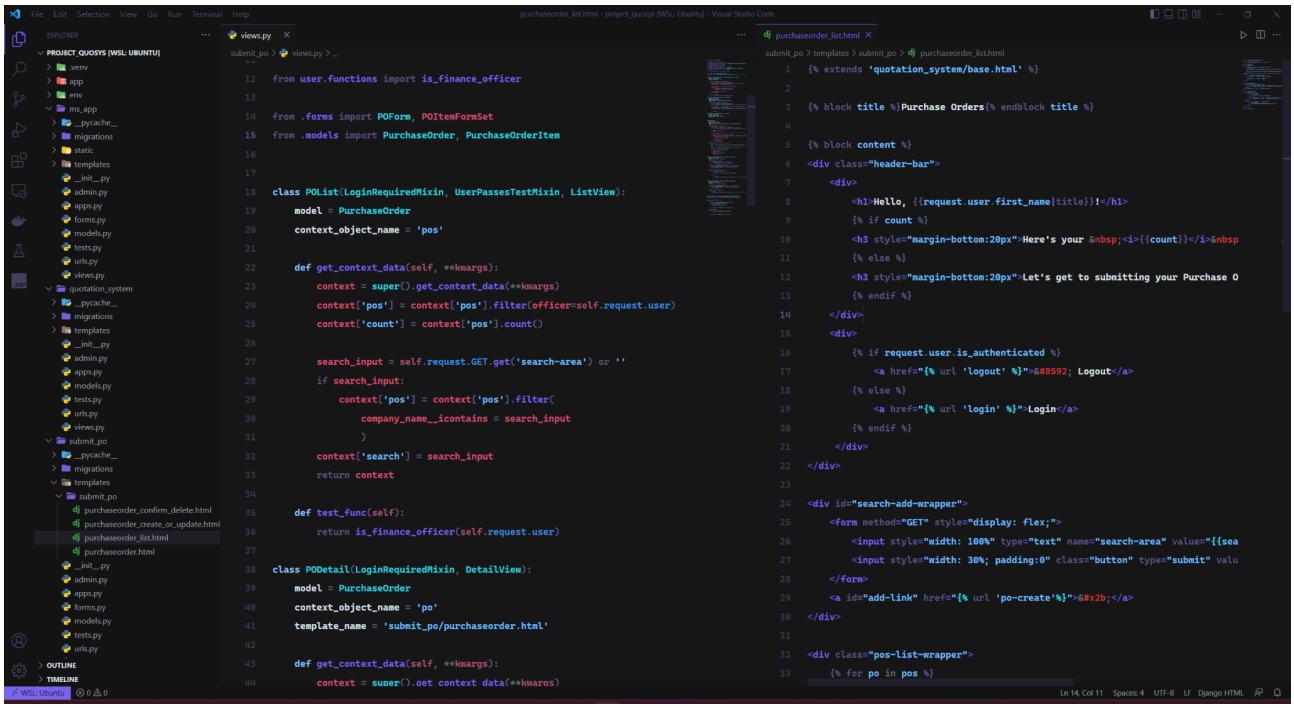
This project was developed on the Windows 11 Operating System. The integrated development environment (IDE) and web framework used for this project are Visual Studio Code 1.74.3 and Django 4.1, respectively.

Use a testing tool, such as the Django Test Framework, for testing. Django has an integrated testing framework that enables you to write unit and integration tests for your application. It may be used to test different application parts (views, models, forms, etc.) to make sure they function as planned.The Django Test Framework enables you to create tests to make sure your application performs as intended, and the Django ORM makes it simple to connect with your database.

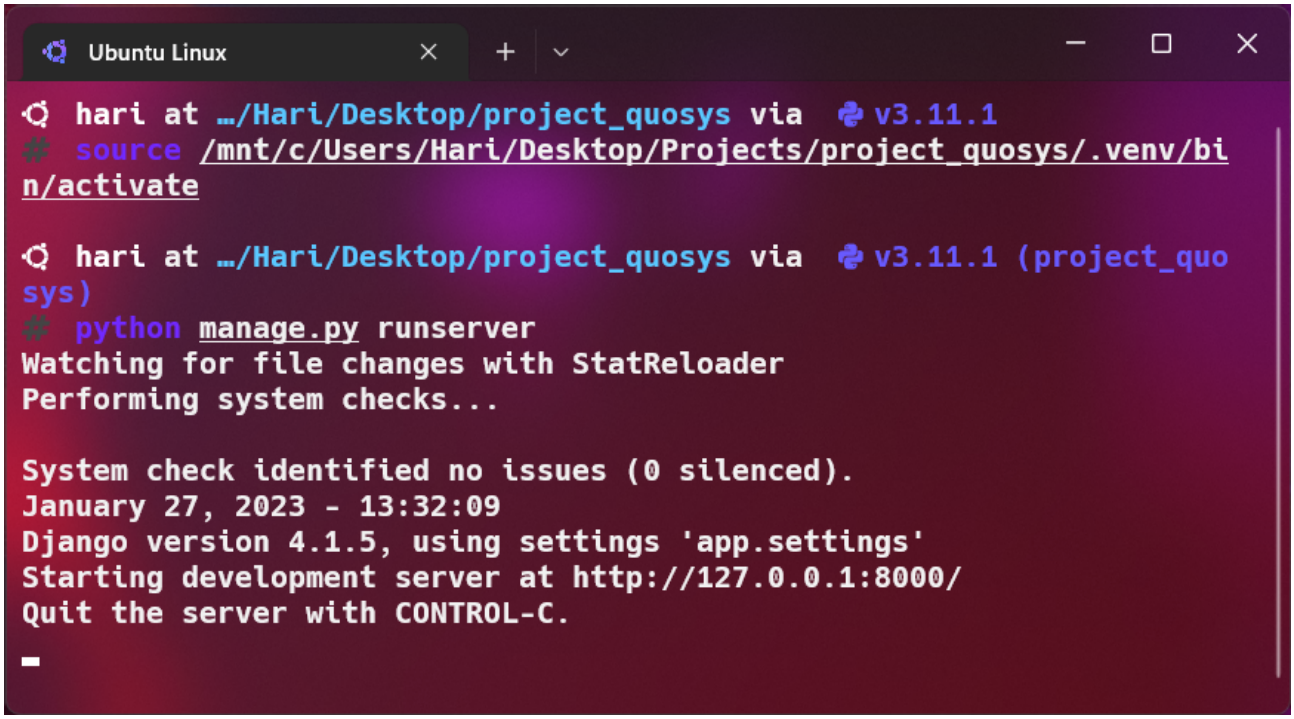
4.1.1 Environment



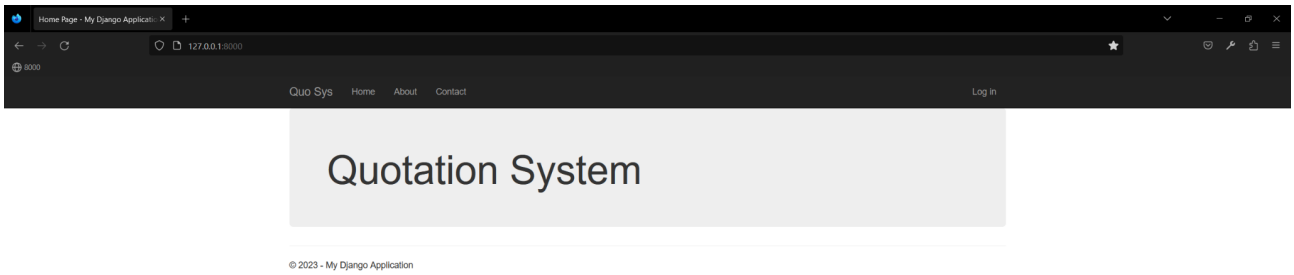
4.1.2 IDE

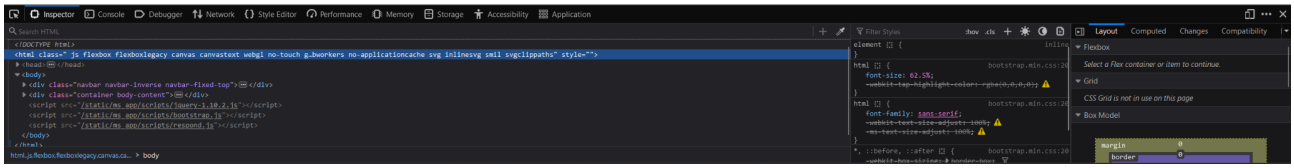


4.1.3 Terminal

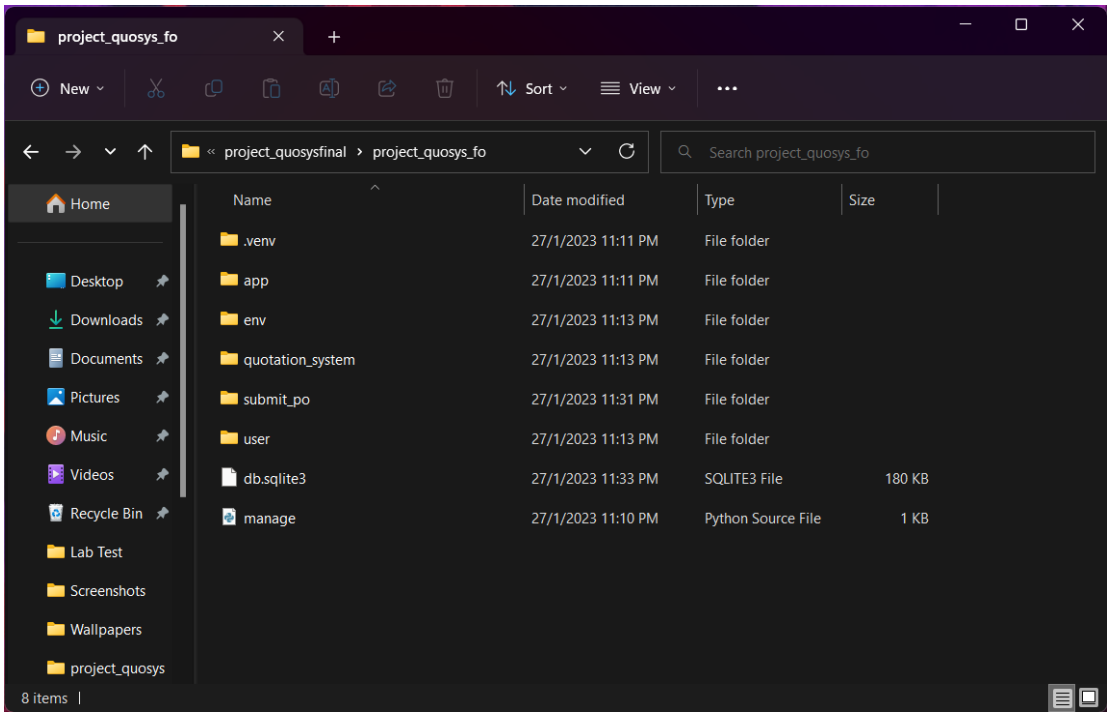


4.1.4 Browser





4.1.5 Project Folder



4.2 Main Program Codes

This component uses Django's Class Based Views to achieve its functionality.

4.2.1 views.py

This is one of the main files to provide the functionality to the views. For each functionality this component has, there is a class based view responsible for it.

View Purchase Order List

```
class POList(LoginRequiredMixin, UserPassesTestMixin, ListView):

    model = PurchaseOrder

    context_object_name = 'pos'


    def get_context_data(self, **kwargs):

        context = super().get_context_data(**kwargs)

        context['pos'] =

context['pos'].filter(officer=self.request.user)

        context['count'] = context['pos'].count()


        search_input = self.request.GET.get('search-area') or ''

        if search_input:

            context['pos'] = context['pos'].filter(

                company_name__icontains = search_input

                )

        context['search'] = search_input

        return context


    def test_func(self):
```

```
        return is_finance_officer(self.request.user)
```

View Purchase Order

```
class PODetail(LoginRequiredMixin, DetailView):

    model = PurchaseOrder

    context_object_name = 'po'

    template_name = 'submit_po/purchaseorder.html'


    def get_context_data(self, **kwargs):

        context = super().get_context_data(**kwargs)

        context['poitems'] =

PurchaseOrderItem.objects.filter(po=context['po'].id)


        context['total'] =

context['poitems'].annotate(total=Sum(F('price') * F('quantity')))

        context['total'] = '{:0.2f}'.format(


(context['total'].aggregate(Sum('total',default=Decimal(0.00)))['tota

l__sum'])

            )


        return context
```

Create and Edit Purchase Order

```
class POCreate(LoginRequiredMixin, POInline, CreateView):
    model = PurchaseOrder
    success_url = reverse_lazy('pos')

    def form_valid(self, form):
        form.instance.officer = self.request.user
        return super(POCreate, self).form_valid(form)

    def get_context_data(self, **kwargs):
        context = super(POCreate, self).get_context_data(**kwargs)
        context['named_formsets'] = self.get_named_formsets()
        return context

    def get_named_formsets(self):
        if self.request.method == "GET":
            return {
                'poitems': POItemFormSet(prefix='poitems'),
            }
        else:
            return {
                'poitems': POItemFormSet(self.request.POST or None,
self.request.FILES or None, prefix='poitems'),
            }
```

```
class POUpdate(LoginRequiredMixin, POInline, UpdateView):
    model = PurchaseOrder
    success_url = reverse_lazy('pos')

    def get_context_data(self, **kwargs):
        context = super(POUpdate, self).get_context_data(**kwargs)
        context['named_formsets'] = self.get_named_formsets()
        return context

    def get_named_formsets(self):
        return {
            'poitems': POItemFormSet(self.request.POST or None,
self.request.FILES or None, instance=self.object, prefix='poitems'),
        }

'''
This 2 functions are for custom added delete button functionality.
If you don't want to use custom delete buttons than don't add this.
'''

def delete_poitem(request, pk):
    try:
        poitem = PurchaseOrderItem.objects.get(id=pk)
    except PurchaseOrderItem.DoesNotExist:
```

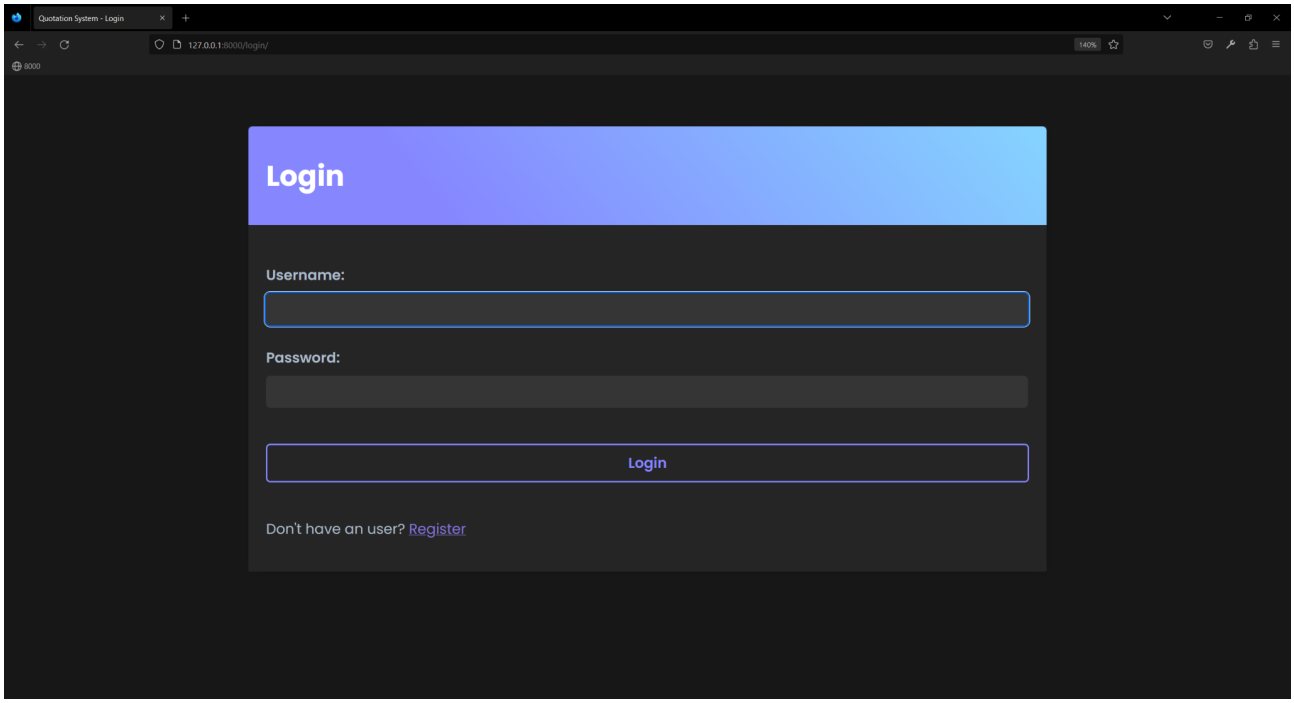
```
        messages.success(
            request, 'Item does not exit'
            )
        return redirect('po-update', pk=poitem.po.id)


    poitem.delete()
    messages.success(
            request, 'Item deleted successfully'
            )


    return redirect('po-update', pk=poitem.po.id)
```

Delete Purchase Order

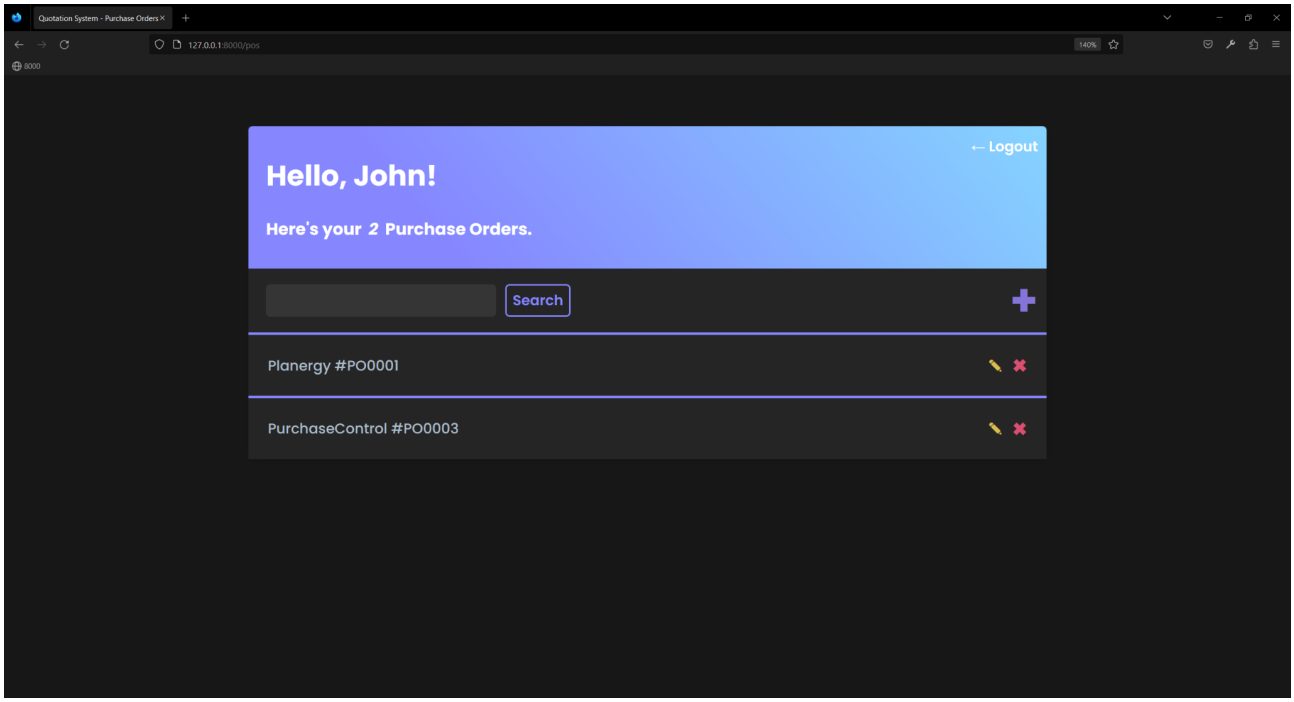
```
class PODelete(LoginRequiredMixin, DeleteView):
    model = PurchaseOrder
    context_object_name = 'po'
    success_url = reverse_lazy('pos')
```

4.3 Sample Screens

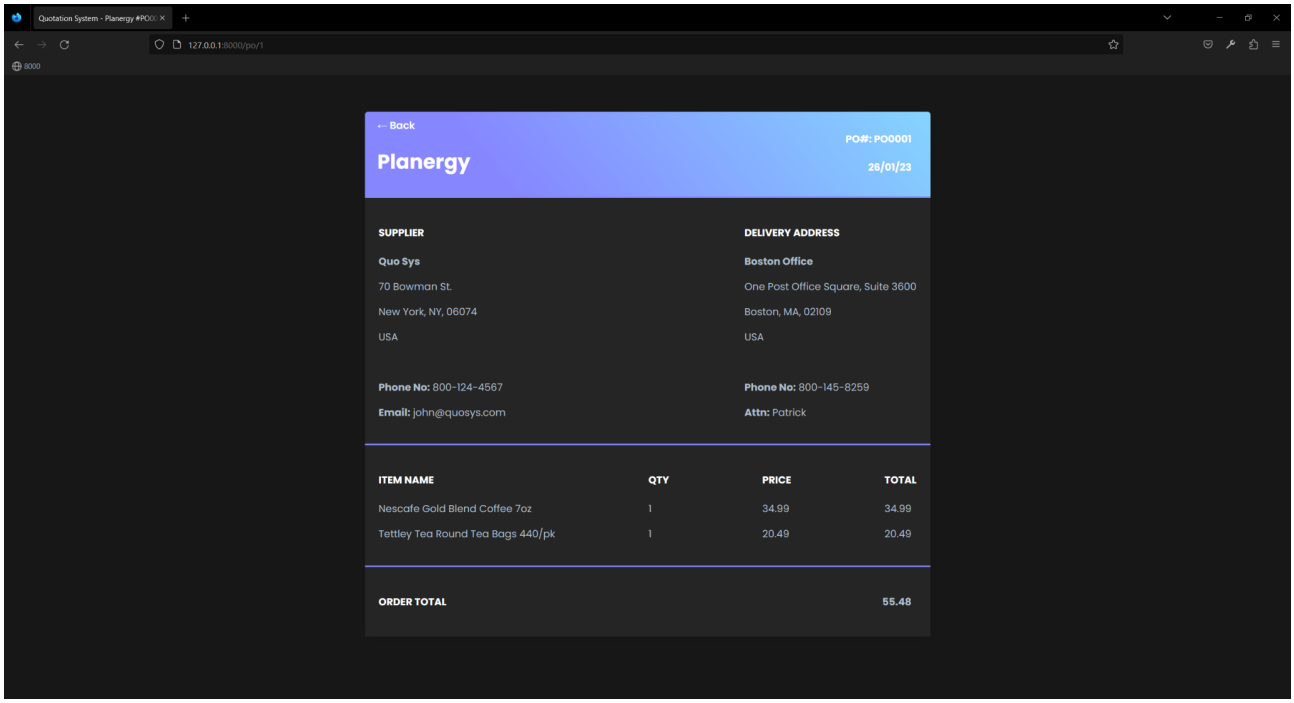
4.3.1 Login



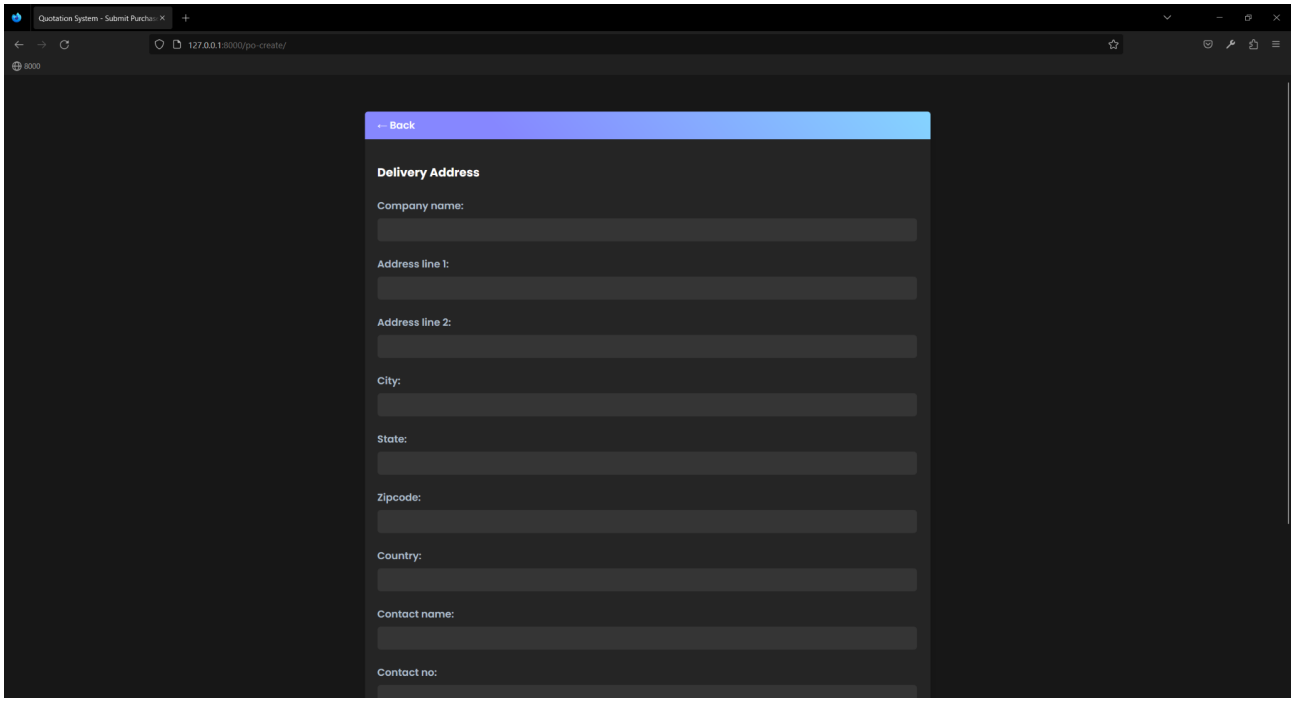
4.3.2 View Purchase Order List

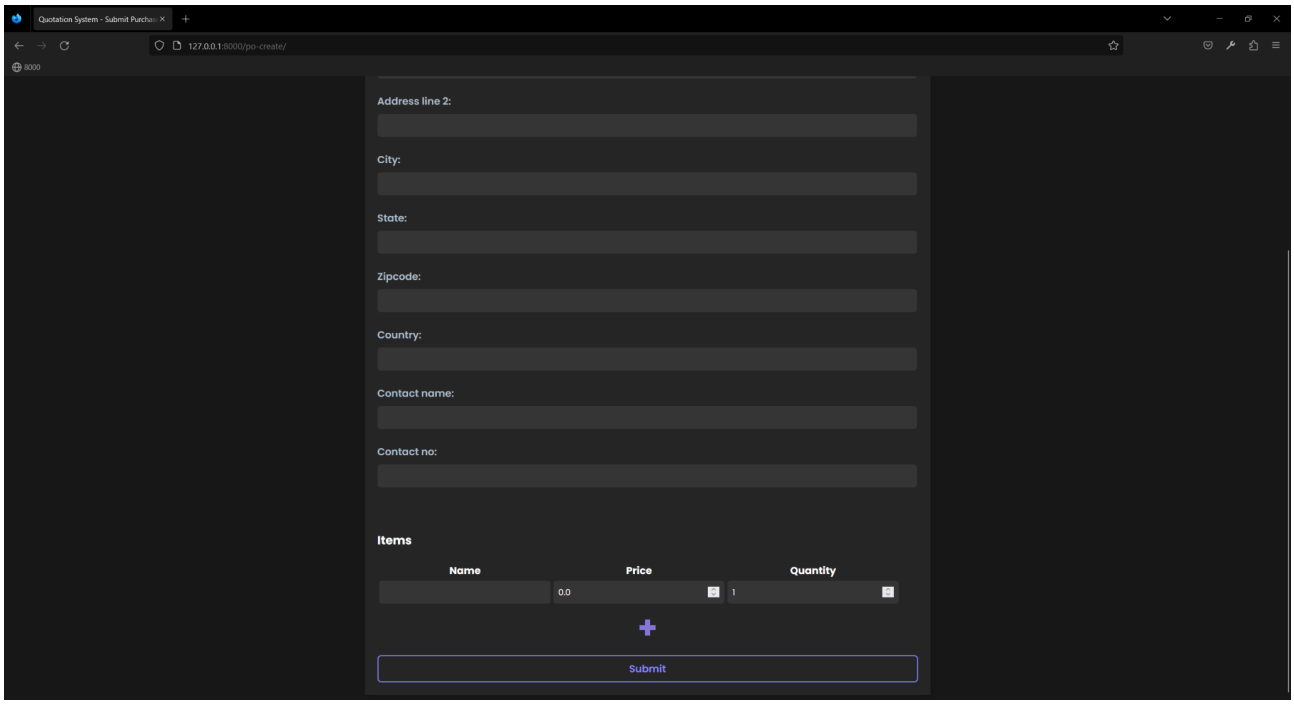


4.3.3 View Purchase Order

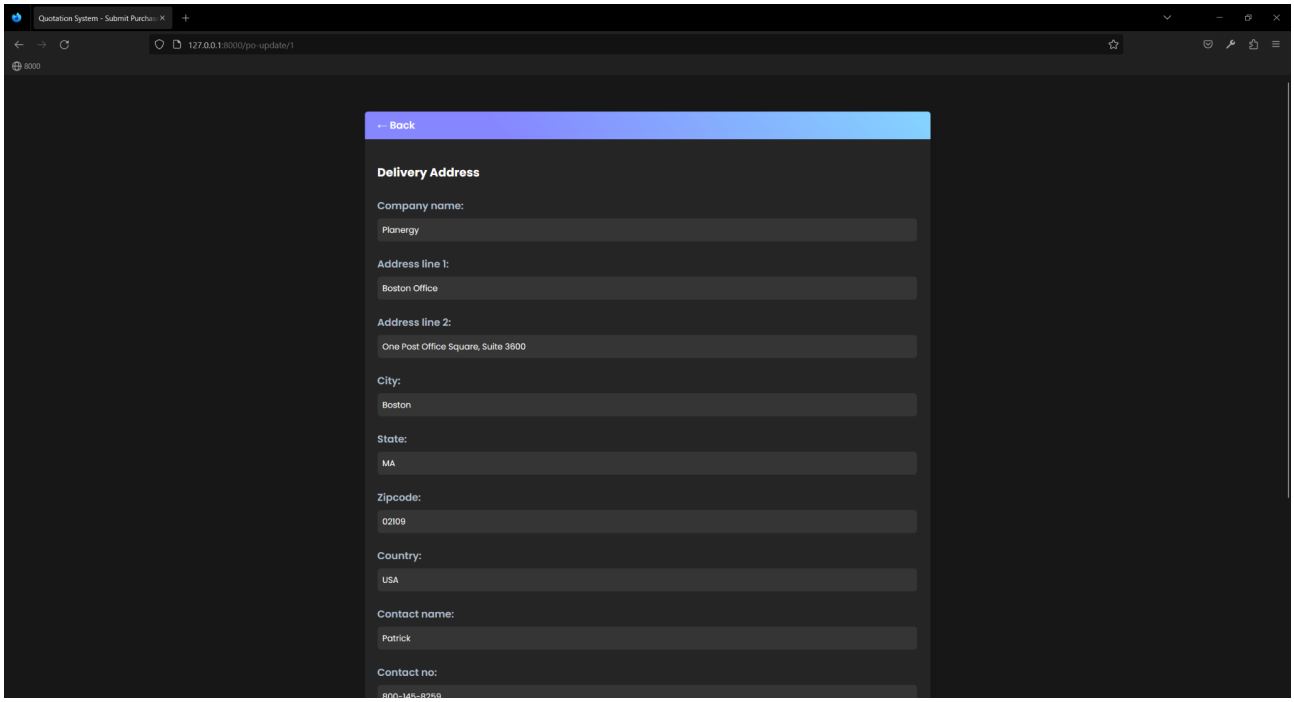


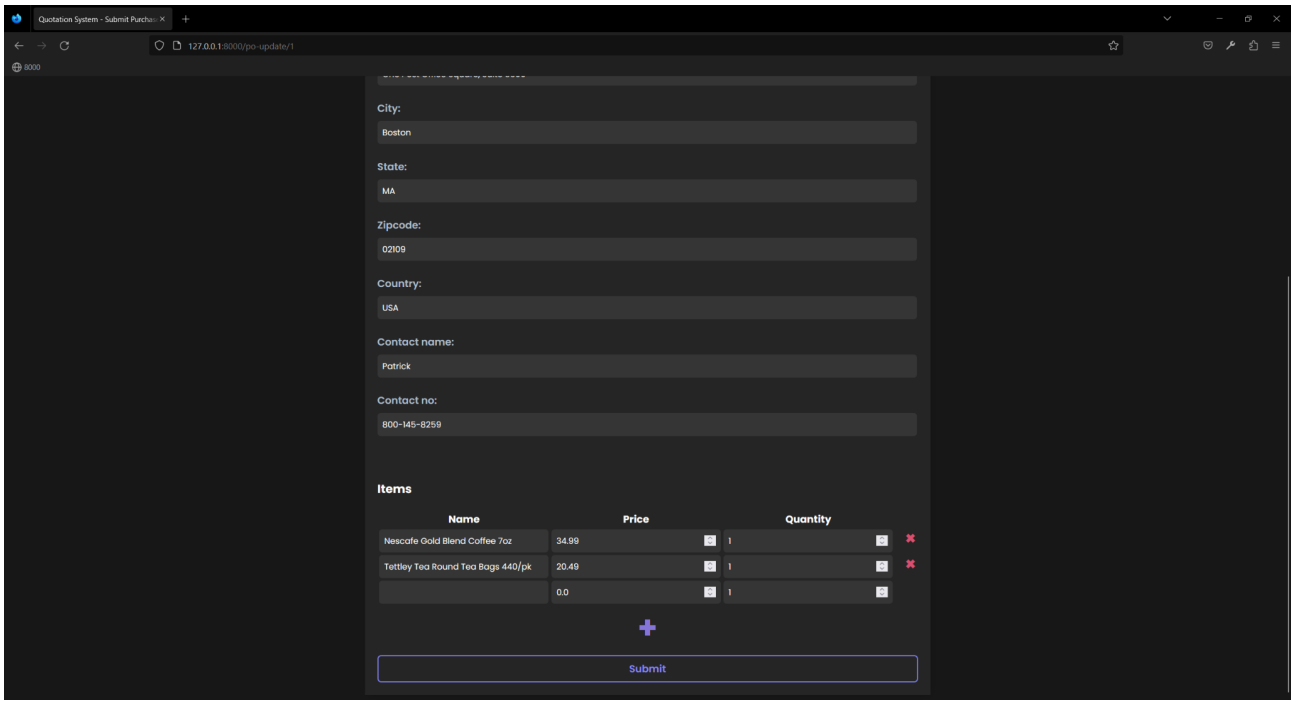
4.3.4 Create Purchase Order



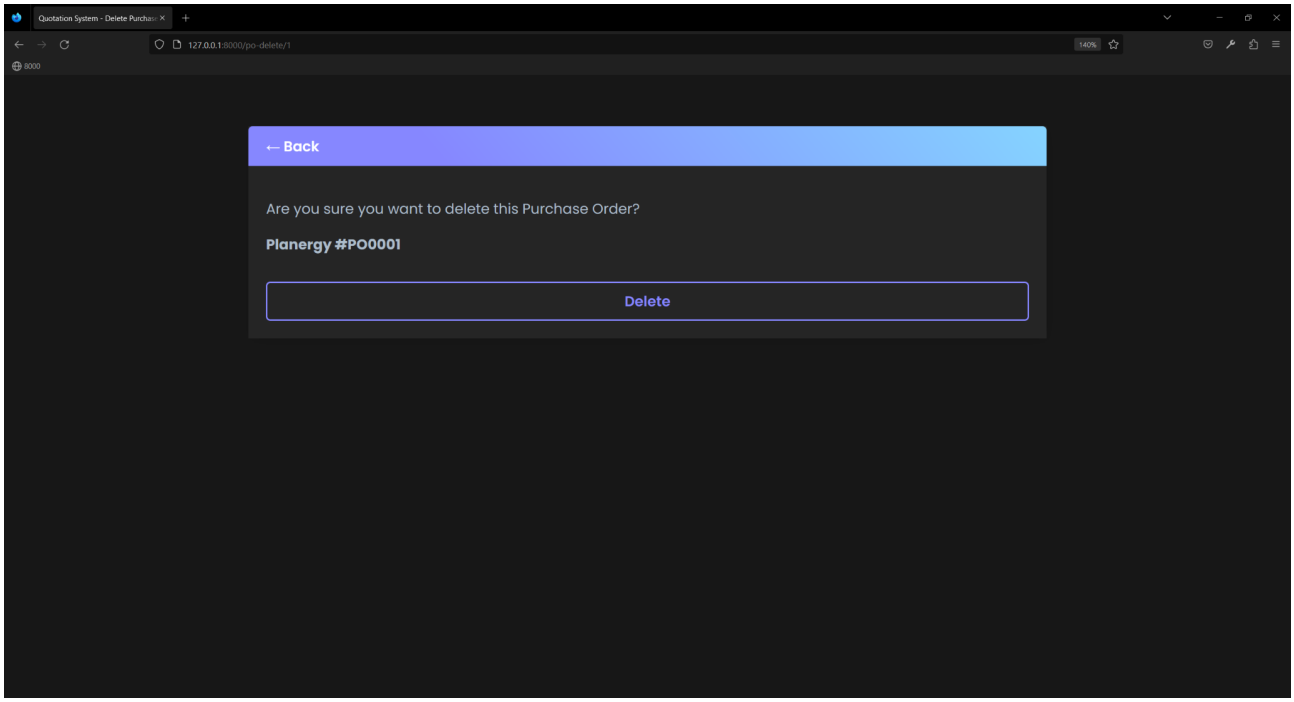


4.3.5 Edit Purchase Order





4.3.6 Delete Purchase Order


Quotation System - Delete Purchase
127.0.0.1:8000/po-delete/1
140%
8000
← Back
Are you sure you want to delete this Purchase Order?
Planergy #PO0001
Delete

5 Testing

5.1 Test Data

Finance Officer

username	password	first_name	last_name	group	email
john-doe	jdjdjdjd	John	Doe	Finance Officer	john@quosys.com

Purchase Order

Officer:	Company name:	Address line 1:	Address line 2:	City:	State:	Country	Zipcode:	Contact name:	Contact no:
john-doe	Planergy	Boston Office	One Post Office Square, Suite 3600	Boston	MA	USA	02109	Patrick	800-145-8259
john-doe	PurchaseControl	California Office	Two Post Office Square, Suite 4800	California	CA	USA	02120	Rick	800-923-4925

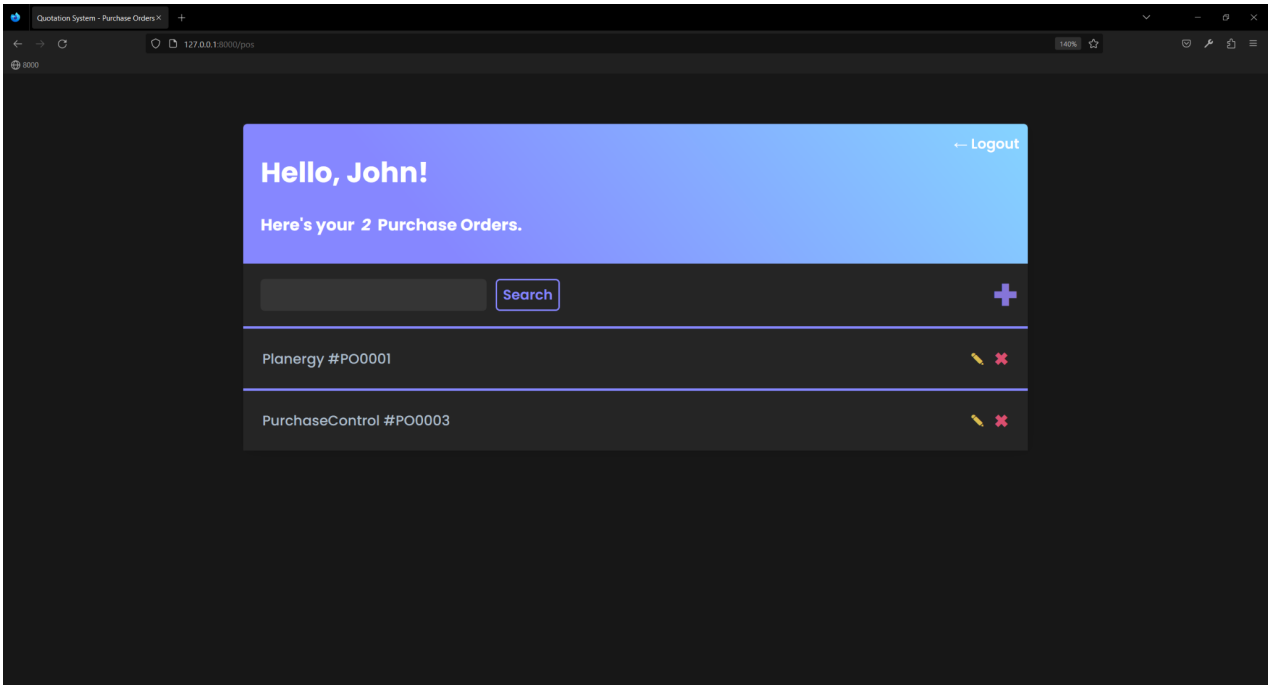
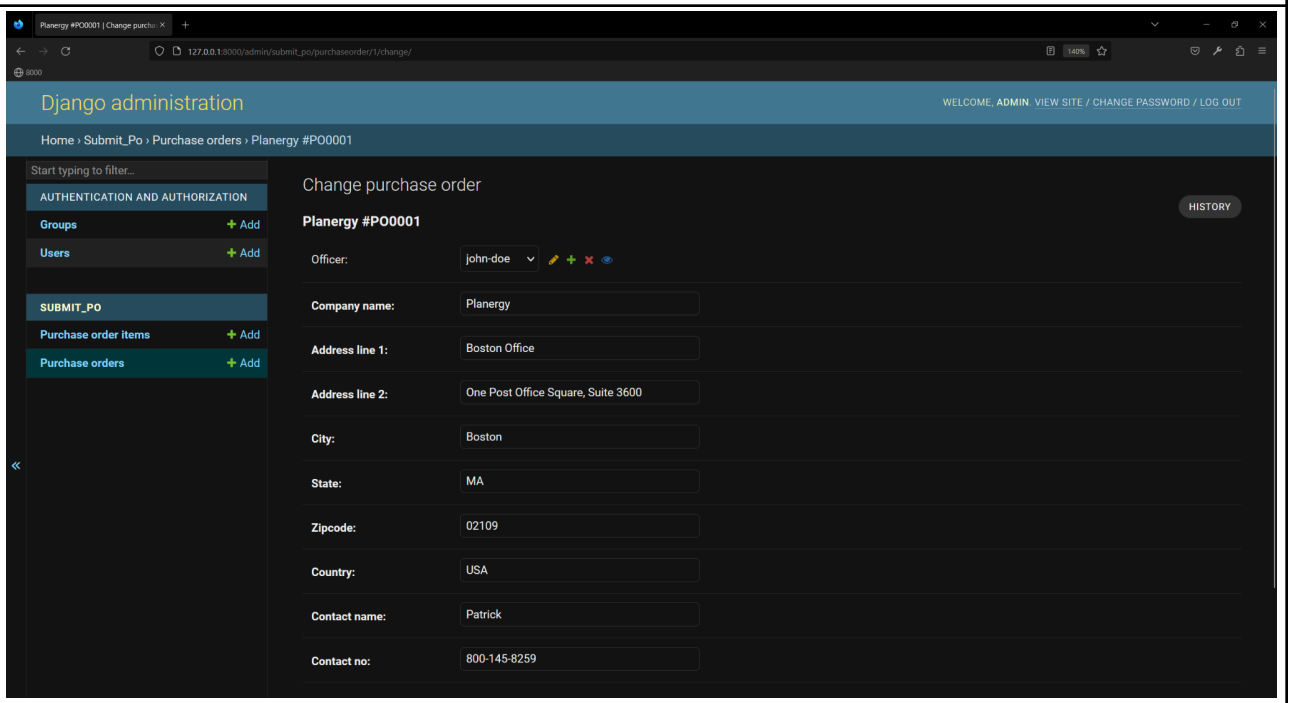
Purchase Order Item

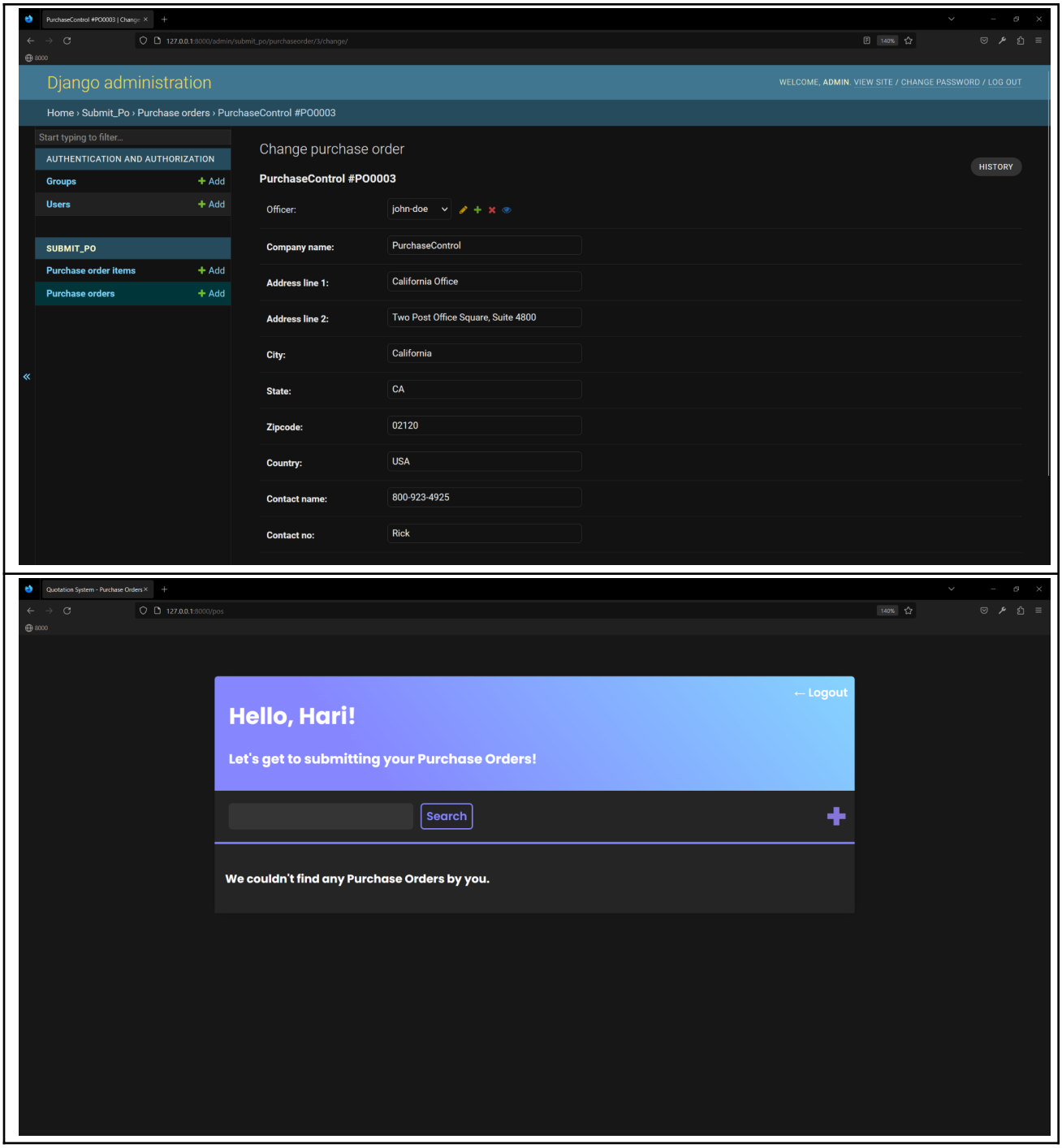
Name	Price	Quantity
Economy Manilla Envelopes - 500	15.49	2
Viking A4 Economy Copier	3.59	5
3 Tier Letter Tray	23.89	3
Tettley Tea Round Tea Bags 440/pk	20.49	1
Nescafe Gold Blend Coffee 7oz	34.99	1

5.2 Acceptance Test

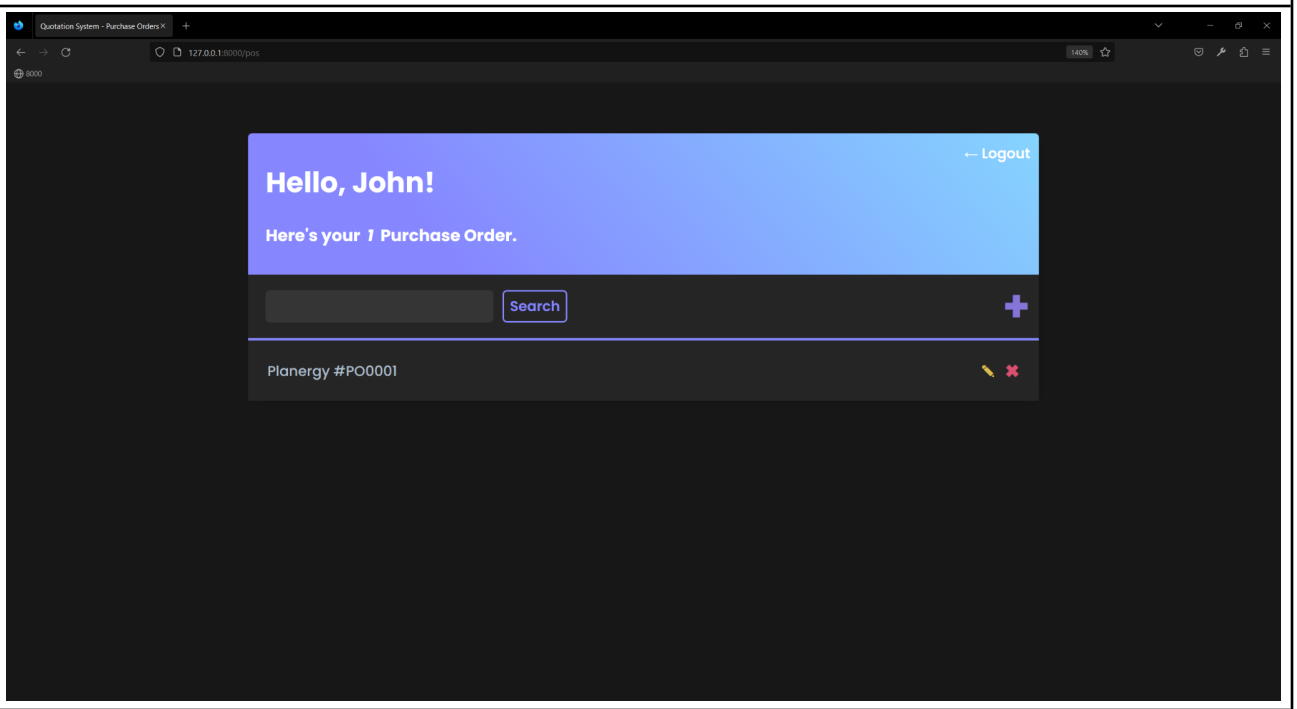
Criteria
The system should display the Finance Officer's Purchase Order Dashboard upon login.
The system should display all key-ins/submissions of Purchase Order by the Finance Officer.
The system should display the option for the Finance Officer to key-in a new Purchase Order to the system.
The system should allow the Finance Officer to key-in the details of a Purchase Order and submit it.

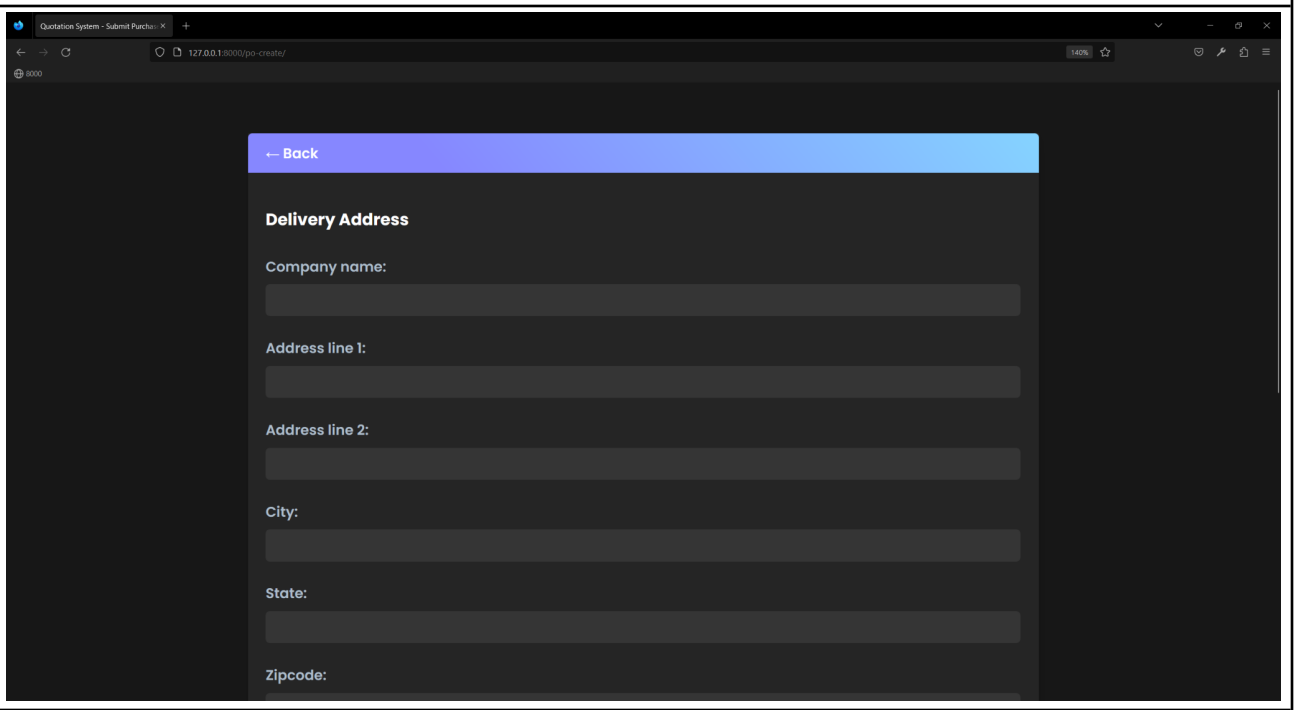
5.3 Test Results

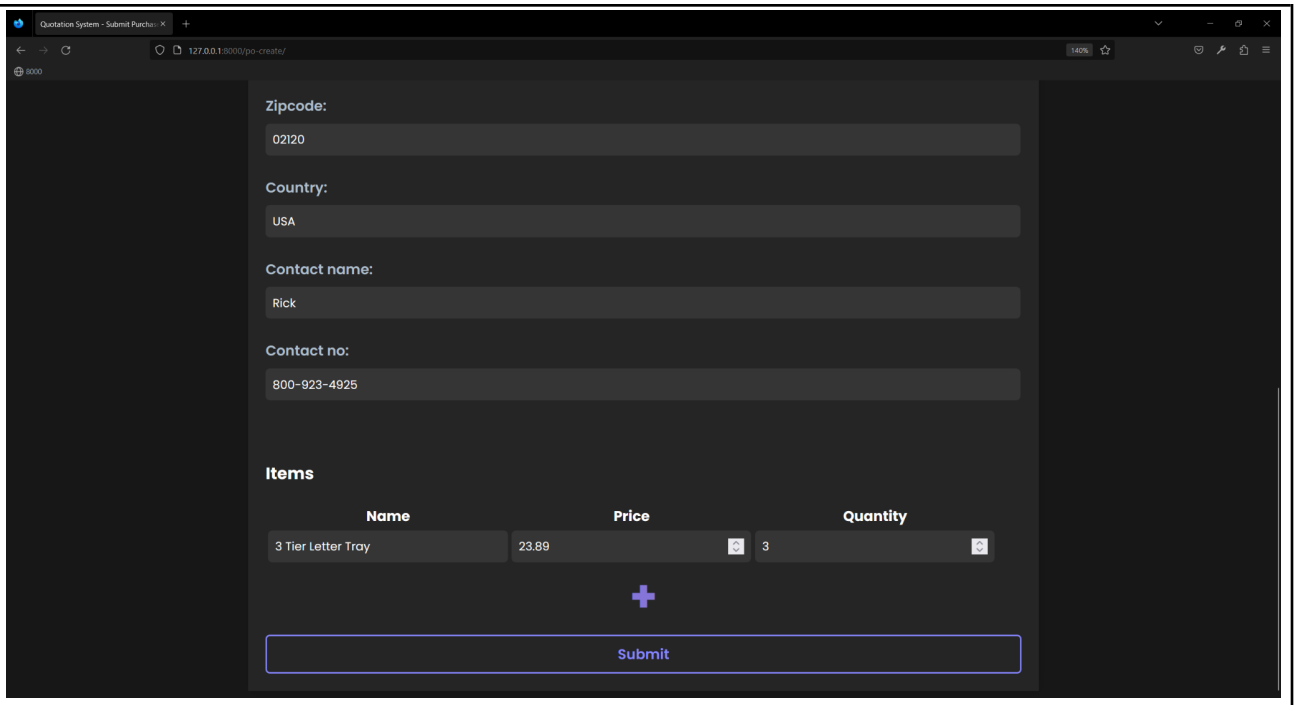
The system displays the Finance Officer's submissions of Purchase Orders. As seen in the Django Admin details attached below. And another Finance Officer's dashboard view.



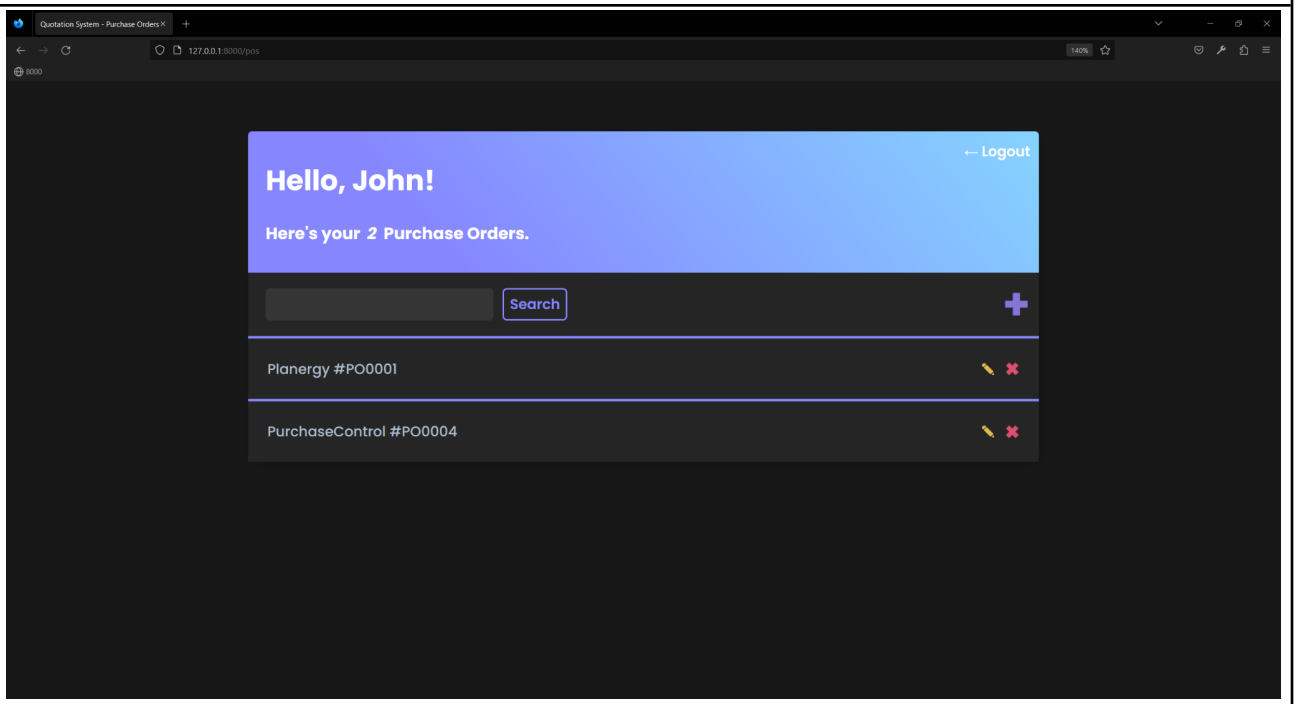

Django administration
WELCOME, ADMIN. VIEW SITE / CHANGE PASSWORD / LOG OUT
Home › Submit_Po › Purchase orders › PurchaseControl #PO0003
Start typing to filter...
AUTHENTICATION AND AUTHORIZATION
Groups + Add
Users + Add
SUBMIT_PO
Purchase order items + Add
Purchase orders + Add
Change purchase order
PurchaseControl #PO0003
HISTORY
Officer: john-doe
Company name: PurchaseControl
Address line 1: California Office
Address line 2: Two Post Office Square, Suite 4800
City: California
State: CA
Zipcode: 02120
Country: USA
Contact name: 800-923-4925
Contact no: Rick
← Logout
Hello, Hari!
Let's get to submitting your Purchase Orders!
Search
We couldn't find any Purchase Orders by you.

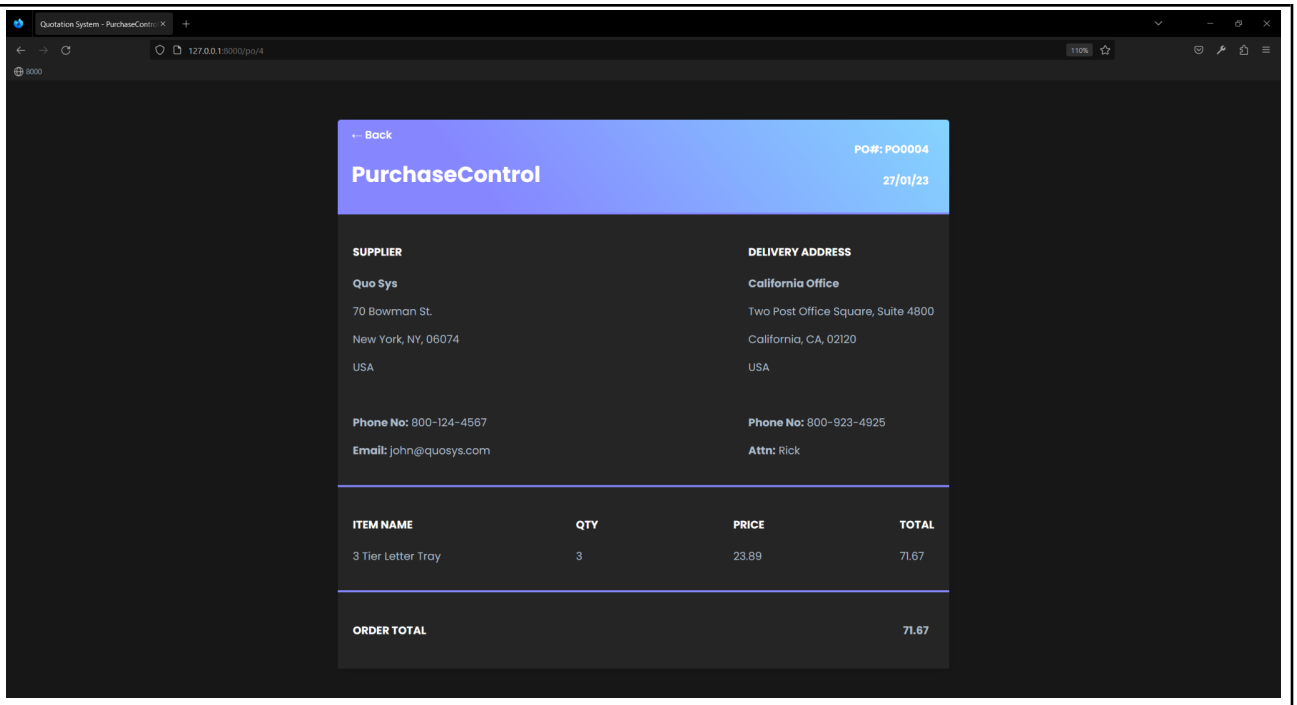
The system should display the option for the Finance Officer to key-in a new Purchase Order to the system. As seen there is a plus icon to add a new Purchase Order. Upon filling up the form and adding the items. There is an option to submit And upon submit, the new Purchase Order is recorded and listed.






Zipcode:
02120
Country:
USA
Contact name:
Rick
Contact no:
800-923-4925
Items
Name
Price
Quantity
3 Tier Letter Tray
23.89
3
Submit


← Logout
Hello, John!
Here's your 2 Purchase Orders.
Search
Planergy #PO0001
PurchaseControl #PO0004


Quotation System - PurchaseControl
127.0.0.1:8000/po/4
110%
8000
← Back
PurchaseControl
PO#: PO0004
27/01/23
SUPPLIER
Quo Sys
70 Bowman St.
New York, NY, 06074
USA
Phone No: 800-124-4567
Email: john@quosys.com
DELIVERY ADDRESS
California Office
Two Post Office Square, Suite 4800
California, CA, 02120
USA
Phone No: 800-923-4925
Attn: Rick
ITEM NAME
QTY
PRICE
TOTAL
3 Tier Letter Tray
3
23.89
71.67
ORDER TOTAL
71.67

6 Conclusion

6.1 Project Achievements

The Finance Officer component of this project was developed in an effective way and with implementations of features that significantly enhanced the user experience, with an user-friendly and aesthetic interface. This component was also successfully developed in accordance with the specifications, and more.

6.2 Quality Assurance

We believe that we have handled the exception handling well and managed to avoid any errors popping up. I identified and corrected defects early in the development process, before they can cause problems for end users. We carried out unit testing and created and maintained a thorough test strategy. We also carried out examinations and tours of software items. One of the critical quality elements I place the most focus on in my software development projects is maintainability. I believe that a project's capacity to easily maintain and upgrade its software is crucial to its long-term success. To ensure maintainability, I use a number of best practices, such as adopting precise and consistent coding guidelines, and segregation of apps and components..

6.3 Problems Encountered

Django was a very straightforward framework to use. However, its simplicity has increased the chances of errors. Lack of time was another obstacle for us to complete this project. Additionally, we used a more agile development methodology, which enabled us to modify the functionality in response to fresh data and user input. Despite our initial lack of time and expertise, this allowed us to stay on schedule for the project. Overall, even though the absence of precise criteria was a serious obstacle, we were able to lessen its effects by putting in place efficient communication and adaptable development procedures.

6.4 Remarks/Comments

I managed to fulfil the requirements of this component for my team and this project, this was a great learning experience dealing with web applications and Django. Django is an all-around strong and adaptable web framework that can be used to create reliable and scalable online applications.