**CHAPTER**-**01**

**INTRODUCTION**

**1.1 Background**

The GST Management System is a Python Django-based web application designed to simplify the process of creating GST-compliant invoices, managing inventory, and maintaining books for businesses. With the introduction of the Goods and Services Tax (GST) in many countries, businesses are required to generate GST-compliant invoices and manage their financial records accordingly. This application addresses the need for a straightforward, user-friendly tool to handle these tasks efficiently.

This system automates invoice generation, tax calculation, and maintains comprehensive records of customers and products. Developed using Python and Django, the system provides a secure, efficient, and scalable solution to modernize billing operations.

**1.2 Objectives and Goals**

**Objective:**

The primary objective of this project is to develop a comprehensive GST billing application using the Django framework that simplifies the processes of invoice creation, inventory management, and bookkeeping for small to medium-sized businesses. The application aims to provide a user-friendly interface, ensure compliance with GST regulations, and streamline business operations by integrating various functionalities into a single platform.

**Goals:**

**1.User-Friendly Interface:**

* **Intuitive Design:** Develop an application with a clean, intuitive interface that can be easily navigated by users with varying levels of technical proficiency.
* **Responsive Design:** Ensure the application is responsive and accessible across different devices, including desktops, tablets, and smartphones.

**2.Efficient Invoice Management:**

* **Easy Invoice Creation:** Allow users to create GST-compliant invoices quickly and efficiently, with fields for all necessary information such as buyer and seller details, product/service descriptions, tax rates, and totals.
* **Invoice Editing and Deletion:** Provide functionality to edit and delete invoices as needed, ensuring data accuracy and flexibility.
* **Automated Calculations:** Implement automated tax calculations to minimize errors and save time for the users.

**3.Comprehensive Inventory Management:**

* **Real-Time Tracking:** Enable real-time tracking of inventory levels, including adding new stock, updating existing stock, and recording sales.
* **Product Management:** Allow users to manage product details such as name, description, price, and stock levels efficiently.
* **Low Stock Alerts:** Implement alert notifications for low stock levels to ensure timely restocking.

**4.Robust Bookkeeping and Financial Management:**

* **Transaction Recording:** Facilitate the recording of all financial transactions, including sales, purchases, expenses, and payments.
* **Balance Tracking:** Provide real-time tracking of account balances to give users a clear view of their financial status.
  + **Financial Reports:** Generate comprehensive financial reports, such as profit and loss statements, balance sheets, and GST reports, to assist in business decision-making and compliance.

**5.Secure User Authentication and Management:**

* + **User Authentication:** Implement secure user authentication mechanisms to protect user data and ensure only authorized access.
  + **Role-Based Access Control:** Provide role-based access control to manage permissions for different user roles (e.g., admin, accountant, staff).
  + **Profile Management:** Allow users to manage their profiles, update personal information, and change passwords.

**6.Compliance with GST Regulations:**

* + **GST Compliance:** Ensure that all invoices and financial reports generated by the application are compliant with current GST regulations.
  + **Tax Rate Management:** Allow users to manage different tax rates applicable to various products and services, ensuring accurate tax calculations.

**7.Scalability and Performance:**

* + **Scalable Architecture:** Design the application architecture to be scalable, accommodating growth in the number of users and transactions without compromising performance.
  + **Efficient Database Management:** Optimize database queries and indexing to ensure efficient data retrieval and storage.

**8.Integration and Extensibility:**

* + **Third-Party Integration:** Facilitate integration with third-party services such as payment gateways, accounting software, and e-commerce platforms.
  + **API Support:** Provide API support to allow other applications to interact with the GST billing system, enabling extensibility and automation.

**9.Customer Support and Documentation:**

* + **Help and Support:** Provide in-app help and support resources, including user guides, FAQs, and contact information for customer support.
  + **Documentation:** Maintain comprehensive documentation for developers and users, detailing the application's features, installation, and usage.

**10.Data Security and Backup:**

* + **Data Encryption:** Implement data encryption mechanisms to protect sensitive user data.
  + **Regular Backups:** Ensure regular data backups to prevent data loss and enable recovery in case of system failures.

**CHAPTER**-**02**

**PROBLEM STATEMENT**

**2.1 Existing System**

**Overview:**

The existing systems for managing GST billing, inventory, and bookkeeping are often fragmented, inefficient, and challenging for small to medium-sized businesses (SMBs) to navigate. These systems typically involve a mix of manual processes, standalone software applications, and complex accounting practices, leading to several operational inefficiencies and compliance challenges.

**Key Issues in Existing Systems:**

**1.** **Manual Processes:**

* **Time-Consuming:** Manual billing and bookkeeping are time-consuming and prone to human errors.
* **Inconsistent Data:** Lack of standardization often results in inconsistent and inaccurate data entry.
* **Limited Accessibility:** Physical records and spreadsheets limit accessibility and sharing of information, hindering collaboration.

**2**.**Fragmented Systems:**

* **Disparate Tools:** Businesses often use separate tools for invoicing, inventory management, and bookkeeping, leading to fragmented data.
* **Integration Challenges:** Difficulty in integrating different systems results in redundant data entry and potential errors.

**3.Compliance Issues:**

* **Complex GST Calculations:** Manual calculation of GST can be complex, especially with varying tax rates and regulations.
* **Regulatory Changes:** Keeping up with frequent changes in GST regulations is challenging without automated updates.

**4.Limited Reporting Capabilities:**

* **Basic Reports:** Many existing systems offer basic reporting capabilities, lacking in-depth financial analysis.
* **Customization Issues:** Limited ability to customize reports according to specific business needs.

**5.Scalability and Performance:**

* **Lack of Scalability:** Existing systems may not scale well with the growth of the business, leading to performance issues.
* **Data Management:** Inefficient data management practices can result in slow processing times and data retrieval issues.

**6.** **Security Concerns:**

* **Data Security:** Manual and fragmented systems pose significant data security risks, including unauthorized access and data breaches.
* **Backup and Recovery:** Lack of systematic data backup and recovery procedures increases the risk of data loss.

**7.** **User Experience:**

* **Complex Interfaces:** Existing software may have complex interfaces that are not user-friendly, especially for non-technical users.
* **Learning Curve:** Steep learning curve for employees to understand and effectively use the software.

**2.2 Proposed System**

**Overview:**

The proposed Django GST billing application addresses the limitations of existing systems by offering an integrated, user-friendly, and scalable solution for GST invoicing, inventory management, and bookkeeping. The system leverages modern web technologies to provide a seamless user experience, automate complex tasks, and ensure compliance with GST regulations.

**Key Features of the Proposed System:**

**1.Integrated Solution:**

* **Unified Platform:** Combines invoicing, inventory management, and bookkeeping into a single platform, reducing redundancy and data fragmentation.
* **Centralized Data:** Ensures all business data is stored centrally, improving data consistency and accessibility.

**2.Automation and Efficiency:**

* **Automated Calculations:** Automates GST calculations, reducing errors and saving time.
* **Inventory Updates:** Automatically updates inventory levels based on sales and purchases, providing real-time tracking.

**3.Compliance and Updates:**

* **GST Compliance:** Ensures all invoices and reports are GST-compliant, with automated updates to accommodate regulatory changes.
* **Tax Rate Management:** Allows easy management of different tax rates for various products and services.

**4.Advanced Reporting:**

* **Comprehensive Reports:** Generates detailed financial reports, including profit and loss statements, balance sheets, and GST reports.
* **Customizable Reports:** Provides options to customize reports based on specific business requirements.

**5.Scalability and Performance:**

* **Scalable Architecture:** Designed to scale with the growth of the business, maintaining performance with increased data and user load.
* **Efficient Data Management:** Utilizes efficient data management practices for quick data retrieval and processing.

**CHAPTER-03**

**SYSTEM REQUIREMENT SPECIFICATION**

**2.3 Functional Requirements**

* **Invoice Creation**: Users should be able to create, edit, and delete invoices.
* **Inventory Management**: Real-time tracking of stock levels, product details, and automatic updates on sales.
* **Bookkeeping**: Recording of financial transactions, tracking balances, and generating financial reports.
* **User Management**: User authentication, role-based access control, and profile management.

**3.2 Software Requirements**

* **Django**: Web framework for developing the application.
* **Python**: Programming language for server-side development.
* **Database**: SQLite (for development), PostgreSQL (for production).
* **HTML/CSS/JavaScript**: Front-end technologies.
* **Bootstrap**: For responsive design.
* **Libraries**: Django REST Framework, Django (for authentication), etc

**3.3 Hardware Requirements**

**•** Computer with multi-core processor (Intel i5 or AMD equivalent).

**•** 4 GB RAM.

**•** 500 GB SSD.

**CHAPTER-04**

**SYSTEM ANYALYSIS AND DESIGN**

**4.1 System Architecture**

**Overview:**

The proposed GST billing application is built on the Django framework and follows a modular, scalable, and secure architecture. The system is divided into several interconnected components, each responsible for specific functionalities. The architecture ensures seamless integration, efficient data management, and a responsive user interface.

**Components:**

**1.User Interface (UI):**

* **Front-End Framework:** Utilizes HTML, CSS, JavaScript, and Bootstrap for a responsive and intuitive user interface.
* **Templates:** Django templates for rendering dynamic content and ensuring consistency across the application.

**2.Application Layer:**

 Django **Views:** Handles user requests and returns appropriate responses, including rendering templates or JSON responses for APIs.

 Forms**:** Manages form submissions and validations, ensuring data integrity.

**3.Business Logic Layer:**

* **Models:** Defines the data structure and relationships, representing the core business entities such as invoices, products, customers, and transactions.
* **Services:** Encapsulates the business logic, including GST calculations, inventory updates, and financial reporting.

**4.Data Access Layer:**

* **Django ORM:** Manages database interactions, providing an abstraction layer for database operations.
* **Database:** Stores all application data, utilizing a relational database like PostgreSQL for robust data management and scalability.

**5.** **Security Layer:**

* **Authentication and Authorization:** Manages user authentication (e.g., login, registration) and authorization (e.g., role-based access control).
* **Data Encryption:** Ensures sensitive data is encrypted both at rest and in transit.

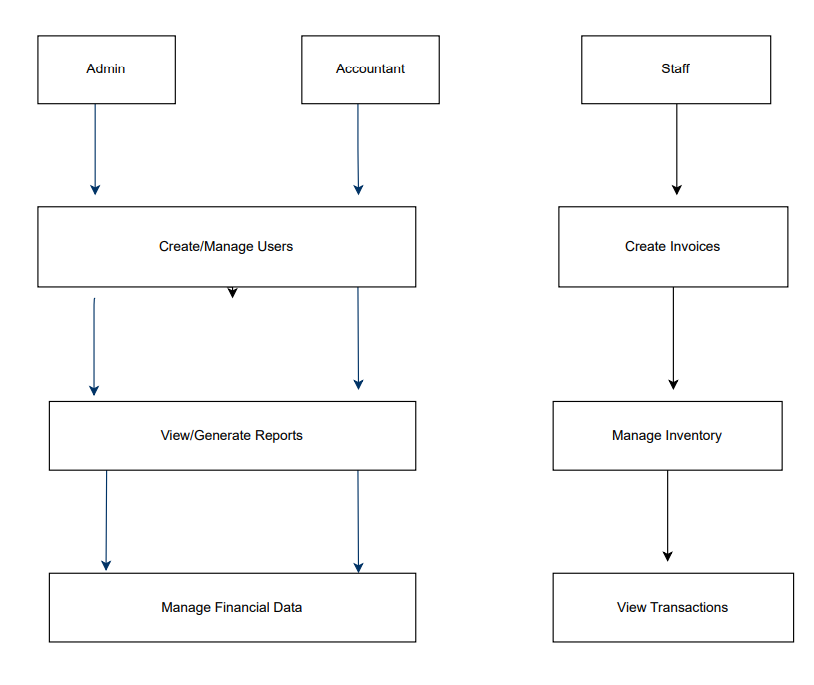
**6**.**Integration Layer:**

* **APIs:** Provides RESTful APIs for integration with third-party services, such as payment gateways and accounting software.
* **Webhooks:** Supports real-time data synchronization with external systems.

**7.Utility Layer:**

* **Logging and Monitoring:** Implements logging for tracking application events and monitoring for performance and security.
* **Backup and Recovery:** Ensures regular data backups and facilitates data recovery in case of system failures.

**4.2 Data Flow Diagram**

****

**Data Flow Diagram**

**CHAPTER-05**

**IMPLEMENTATION**

This chapter is to show the code involved in the creation of frontend interfaces and the code responsible for connecting these interfaces and the underlying database.

**5.1 HTML/CSS (Front end)**

**Python code to connect to Backend**

import datetime

import json

import num2words

from django.shortcuts import render

from django.shortcuts import redirect

from django.shortcuts import get\_object\_or\_404

from django.http import HttpResponse

from django.http import JsonResponse

from django.db.models import Max

from django.contrib.auth.decorators import login\_required

from django.contrib.auth import login

from django.contrib.auth.forms import AuthenticationForm, UserCreationForm

from. models import Customer

from. models import Invoice

from. models import Product

from. models import UserProfile

from. models import Inventory

from. models import InventoryLog

from. models import Book

from. models import BookLog

from. utils import invoice\_data\_validator

from. utils import invoice\_data\_processor

from. utils import update\_products\_from\_invoice

from. utils import update\_inventory

from. utils import create\_inventory

from. utils import add\_customer\_book

from. utils import auto\_deduct\_book\_from\_invoice

from. utils import remove\_inventory\_entries\_for\_invoice

from. forms import CustomerForm

from. forms import ProductForm

from. forms import UserProfileForm

from. forms import InventoryLogForm

from. forms import BookLogForm

# Create your views here.

# User Management

@login\_required

def user\_profile\_edit(request):

context = {}

user\_profile = get\_object\_or\_404(UserProfile, user=request.user)

context['user\_profile\_form'] = UserProfileForm(instance=user\_profile)

if request.method == "POST":

user\_profile\_form = UserProfileForm(request.POST, instance=user\_profile)

user\_profile\_form.save()

return redirect('user\_profile')

return render (request, 'gstbillingapp/user\_profile\_edit.html', context)

@login\_required

def user\_profile(request):

context = {}

user\_profile = get\_object\_or\_404(UserProfile, user=request.user)

context['user\_profile'] = user\_profile

return render (request, 'gstbillingapp/user\_profile.html', context)

def login\_view(request):

if request.user.is\_authenticated:

return redirect("invoice\_create")

context = {}

auth\_form = AuthenticationForm(request)

if request.method == "POST":

auth\_form = AuthenticationForm(request, data=request.POST)

if auth\_form.is\_valid():

user = auth\_form.get\_user()

if user:

login (request, user)

return redirect("invoice\_create")

else:

context["error\_message"] = auth\_form.get\_invalid\_login\_error()

context["auth\_form"] = auth\_form

return render (request, 'gstbillingapp/login.html', context)

def signup\_view(request):

if request.user.is\_authenticated:

return redirect("invoice\_create")

context = {}

signup\_form = UserCreationForm()

profile\_edit\_form = UserProfileForm()

context["signup\_form"] = signup\_form

context["profile\_edit\_form"] = profile\_edit\_form

if request.method == "POST":

signup\_form = UserCreationForm(request.POST)

profile\_edit\_form = UserProfileForm(request.POST)

context["signup\_form"] = signup\_form

context["profile\_edit\_form"] = profile\_edit\_form

if signup\_form.is\_valid():

user = signup\_form.save()

else:

context["error\_message"] = signup\_form.errors

return render (request, 'gstbillingapp/signup.html', context)

if profile\_edit\_form.is\_valid():

userprofile = profile\_edit\_form.save(commit=False)

userprofile.user = user

userprofile.save()

login (request, user, backend='django.contrib.auth.backends.ModelBackend')

return redirect("invoice\_create")

return render (request, 'gstbillingapp/signup.html', context)

# Invoice, products and customers

@login\_required

def invoice\_create(request):

# if business info is blank redirect to update it

user\_profile = get\_object\_or\_404(UserProfile, user=request.user)

if not user\_profile.business\_title:

return redirect('user\_profile\_edit')

context = {}

context['default\_invoice\_number'] = Invoice.objects. filter(user=request.user).aggregate(Max('invoice\_number'))['invoice\_number\_\_max']

if not context['default\_invoice\_number']:

context['default\_invoice\_number'] = 1

else:

context['default\_invoice\_number'] += 1

context['default\_invoice\_date'] = datetime.datetime.strftime(datetime.datetime.now(), '%Y-%m-%d')

if request.method == 'POST':

print ("POST received - Invoice Data")

nvoice\_data = request.POST

validation\_error = invoice\_data\_validator(invoice\_data)

if validation\_error:

context["error\_message"] = validation\_error

return render (request, 'gstbillingapp/invoice\_create.html', context)

# valid invoice data

print ("Valid Invoice Data")

invoice\_data\_processed = invoice\_data\_processor(invoice\_data)

# Save customer

customer = None

try:

customer = Customer.objects.get(user=request.user,

customer\_name=invoice\_data['customer-name'],

customer\_address=invoice\_data['customer-address'],

customer\_phone=invoice\_data['customer-phone'],

customer\_gst=invoice\_data['customer-gst'])

except:

print("===============> customer not found")

print(invoice\_data['customer-name'])

print(invoice\_data['customer-address'])

print(invoice\_data['customer-phone'])

print(invoice\_data['customer-gst'])

if not customer:

print("CREATING CUSTOMER===============>")

customer = Customer(user=request.user,

customer\_name=invoice\_data['customer-name'],

customer\_address=invoice\_data['customer-address'],

customer\_phone=invoice\_data['customer-phone'],

customer\_gst=invoice\_data['customer-gst'])

# create customer book

customer.save()

add\_customer\_book(customer)

# save product

update\_products\_from\_invoice(invoice\_data\_processed, request)

# save invoice

invoice\_data\_processed\_json = json.dumps(invoice\_data\_processed)

new\_invoice = Invoice(user=request.user,

invoice\_number=int(invoice\_data['invoice-number']),

invoice\_date=datetime. datetime.strptime(invoice\_data['invoice-date'], '%Y-%m-%d'),

invoice\_customer=customer, invoice\_json=invoice\_data\_processed\_json)

new\_invoice.save()

print("INVOICE SAVED")

update\_inventory(new\_invoice, request)

print("INVENTORY UPDATED")

auto\_deduct\_book\_from\_invoice(new\_invoice)

print("CUSTOMER BOOK UPDATED")

return redirect('invoice\_viewer', invoice\_id=new\_invoice.id)

return render(request, 'gstbillingapp/invoice\_create.html', context)

**Login page**

{% extends "gstbillingapp/base.html" %}

{% load static %}

{% block content %}

<div class="row">

<div class="col-sm">

</div>

<div class="col-sm">

<div class="card" id="login-card">

<h5 class="card-header">Login</h5>

<div class="card-body">

<h5 class="card-title">Login and start billing now! </h5>

<! -- <p class="card-text">With supporting text below as a natural lead-in to additional content. </p> -->

<! --a href="{% url "social:begin""google-oauth2" %}" class="btn btn-primary">Login with Google</a-->

</div>

<div>OR</div>

<form method="POST">

{% csrf\_token %}

<table class="table">

<tbody class="two-col-form">

<tr>

<th scope="row" class="text-right">Username:</th>

<td class="form-input-td">{{auth\_form.username}}</td>

</tr>`

<tr>

<th scope="row" class="text-right">Password:</th>

<td class="form-input-td">{{auth\_form.password}}</td>

</tr>

</tbody>

</table>

<input type="submit" class="btn btn-primary float-right" value="Submit">

</form>

</div>

<a href="{% url 'signup\_view' %}"><input class="btn btn-danger float-right" value="Sign Up"></a>

</div>

<div class="col-sm">

</div>

</div>

{% endblock %}

**Landing Page**

{% extends "gstbillingapp/base.html" %}

{% load static %}

<!doctype html>

{% block title %}

DjangoGst - Simplest GST invoicing | Inventory Management | Book keeping

{% endblock %}

{% block metatags %}

<meta name="description" content="Create GST invoice/bills for free. Automatic inventory Management and Books management.">

<metaname="google-site-verification" content="MJbX1a9Fs1MIE1tnpRnp4At9OzVMxoumTAvOWUlG8-w" />

<link rel="canonical" href="https://billgst.in/" />

<meta property="og:title" content="BillGST - Simplest GST invoicing | Inventory Management | Book keeping" />

<meta property="og:url" content="https://billgst.in" />

<metaproperty="og:image"content="https://billgst.in{%static "gstbillingapp/images/screenshot\_interface.png" %}" />

{% endblock %}

{% block content %}

<div class="jumbotron">

<h1 class="display-3">GST Management System</h1>

<h2 class="display-6">Simplest <b>GST invoicing</b> app.</h2>

<hr class="my-4">

<h4>

<ul>

<li>Easily create invoices</li>

<li>Manage inventory</li>

<li>Keep books and track balances</li>

</ul>

</h4>

<p class="lead">

<a class="btn btn-primary btn-lg" href="{% url 'invoice\_create' %}" role="button">Create Invoice</a>

</p>

</div>

<div class="row">

<div class="col d-flex align-items-stretch">

<div class="card text-white bg-primary mb-3">

<div class="card-body">

<h4 class="card-title">Simple Interface</h4>

<imgclass="img-fluid"src="{%static"gstbillingapp/images/screenshot\_interface.png" %}" alt="Simple Interface">

</div>

</div>

</div>

<div class="col d-flex align-items-stretch">

<div class="card text-white bg-primary mb-3">

<div class="card-body">

<h4 class="card-title">Customer and Product Search</h4>

<imgclass="img-fluid"src="{%static "gstbillingapp/images/screenshot\_search.png" %}" alt="Customer and Product Search">

</div>

</div>

</div>

<! -- <div class="col d-flex align-items-stretch">

<div class="card text-white bg-primary mb-3">

<div class="card-body">

<h4 class="card-title">Printable Invoices</h4>

<ul>

<li><h5>Clean Invoice Format</h5></li>

</ul>

<ul>

<li><h5>Print Invoices directly with one click.</h5></li>

</ul>

<ul>

<li><h5>Save Invoices as PDF</h5></li>

</ul>

</div>

</div>

</div>

<div class="col d-flex align-items-stretch">

<div class="card text-white bg-primary mb-3">

<div class="card-body">

<h4 class="card-title">Manage Books and Balances</h4>

<imgclass="img-fluid"src="{%static "gstbillingapp/images/screenshot\_books.png" %}" alt="Manage books and balances">

</div>

</div>

</div>

<div class="col d-flex align-items-stretch">

<div class="card text-white bg-primary mb-3">

<div class="card-body">

<h4 class="card-title">Easy Inventory Management</h4>

<imgclass="img-fluid"src="{%static "gstbillingapp/images/screenshot\_inventory.png" %}" alt="Manage inventory easily">

</div>

</div>

</div>

</div>

</div>

{% endblock %}

**Signup Page**

{% extends "gstbillingapp/base.html" %}

{% load static %}

{% block content %}

<h2>Sign Up</h2>

<form method="POST">

{% csrf\_token %}

<table class="table">

<tbody class="two-col-form">

<tr>

<th scope="row" class="text-right">Username:</th>

<td class="form-input-td">{{signup\_form.username}}</td>

</tr>

<tr>

<th scope="row" class="text-right">Password:</th>

<td class="form-input-td">{{signup\_form.password1}}</td>

</tr>

<tr>

<th scope="row" class="text-right">Confirm Password:</th>

<td class="form-input-td">{{signup\_form. password2}}</td>

</tr>

<tr>

<th scope="row" class="text-right">Business Name:</th>

<td class="form-input-td">{{profile\_edit\_form.business\_title}}</td>

</tr>

<tr>

<th scope="row" class="text-right">Address:</th>

<td class="form-input-td">{{profile\_edit\_form. business\_address}}</td>

</tr>

<tr>

<th scope="row" class="text-right">Email:</th>

<td class="form-input-td">{{profile\_edit\_form.business\_email}}</td>

</tr>

<tr>

<th scope="row" class="text-right">Phone:</th>

<td class="form-input-td">{{profile\_edit\_form.business\_phone}}</td>

</tr>

<tr>

<th scope="row" class="text-right">GST:</th>

<td class="form-input-td" pattern="([A-Za-z0-9]{15})|(^$)">{{profile\_edit\_form.business\_gst}}</td>

</tr>

</tbody>

</table>

<input type="submit" class="btn btn-primary float-right" value="Submit">

</form>

{% endblock %}

**Inventory Page**

{% extends "gstbillingapp/base.html" %}

{% load static %}

{% block content %}

<div class="row">

<div class="col">

<h2>Inventory</h2>

</div>

</div>

<table class="table table-hover font-weight-bold" id="inventory-table">

<thead>

<tr>

<th>Product</th>

<th>Current Stock</th>

<th>Last Transaction</th>

</tr>

</thead>

<tbody class="two-col-form">

{% for inventory in inventory\_list %}

<tr onclick="window.location='{% url 'inventory\_logs' inventory.id %}';">

<td><a href="{% url 'inventory\_logs' inventory.id %}">{{inventory.product.product\_name}}</a></td>

<td>{{inventory.current\_stock}}</td>

<td>{{inventory.last\_log}}</td>

</tr>

{% endfor %}

</tbody>

</table>

{% endblock %}

{% block includejs %}

<script type="text/javascript">

$(document).ready( function () {

$('#inventory-table').DataTable({

"order": [],

"columnDefs": [ {

"targets": 3,

"sortable": false,

"searchable": false

} ],

"paging": true

});

});

</script>

{% endblock %}

**User Page**

{% extends "gstbillingapp/base.html" %}

{% load static %}

{% block content %}

<div class="row">

<div class="col">

<h2>Profile</h2>

</div>

<div class="col">

<a href="{% url 'user\_profile\_edit' %}" id="user\_profile\_edit\_btn"><button type="button" class="btn btn-primary">Edit</button></a>

</div>

</div>

<table class="table">

<tbody class="two-col-form">

<tr>

<th scope="row" class="text-right">Business Name:</th>

<td class="form-input-td customer\_search\_area">{{user\_profile.business\_title}}</td>

</tr>

<tr>

<th scope="row" class="text-right">Address:</th>

<td class="form-input-td">{{user\_profile.business\_address}}</td>

</tr>

<tr>

<th scope="row" class="text-right">Email:</th>

<td class="form-input-td">{{user\_profile.business\_email}}</td>

</tr>

<tr>

<th scope="row" class="text-right">Phone:</th>

<td class="form-input-td">{{user\_profile.business\_phone}}</td>

</tr>

<tr>

<th scope="row" class="text-right">GST:</th>

<td class="form-input-td">{{user\_profile.business\_gst}}</td>

</tr>

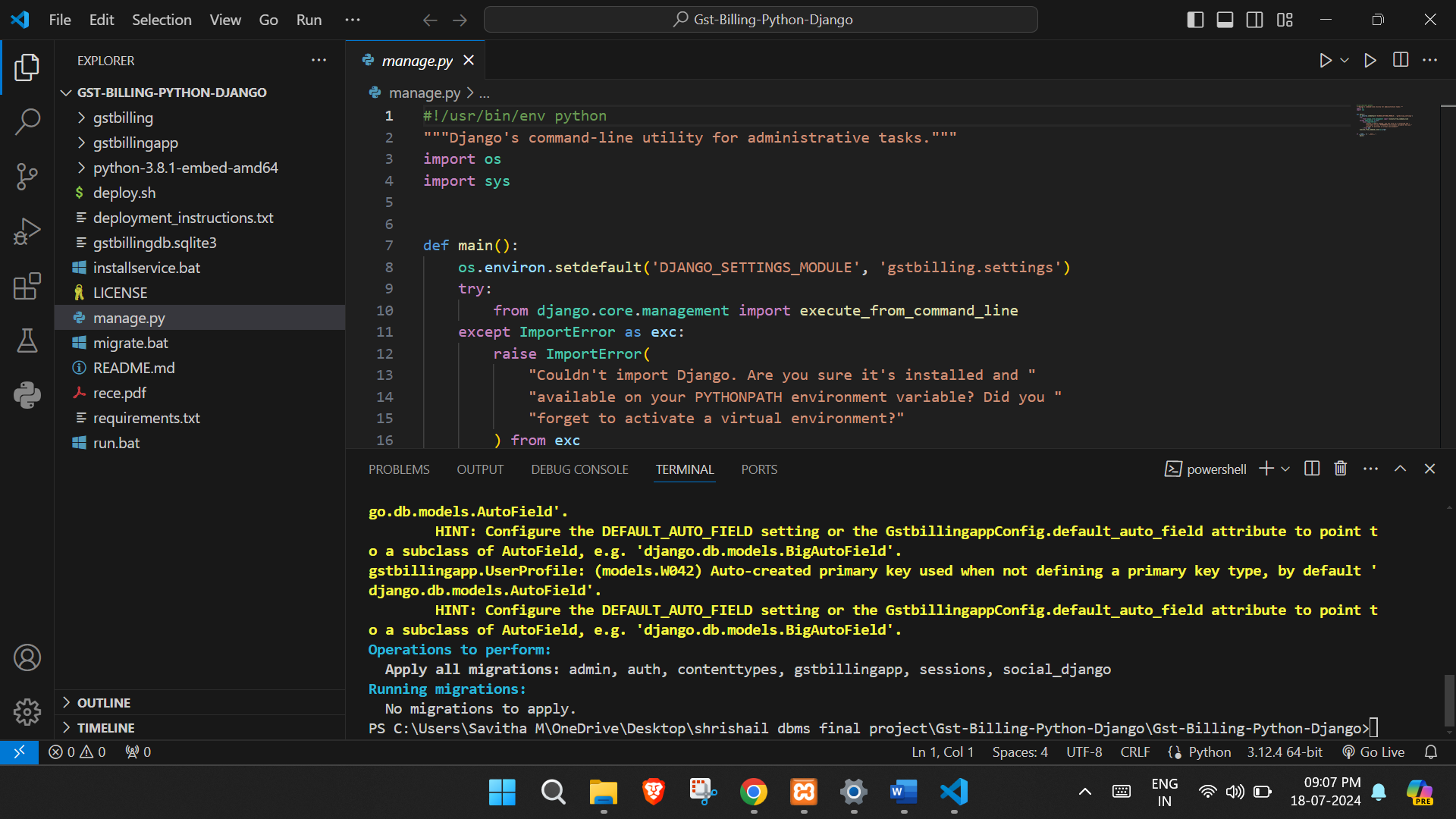
</tbody>

</table>

{% endblock %}

**CHAPTER-05**

**RESULTS AND DISCUSSIONS**

** SNAPSHOTS**

Using Terminal Running Manage.py file to Open the Project

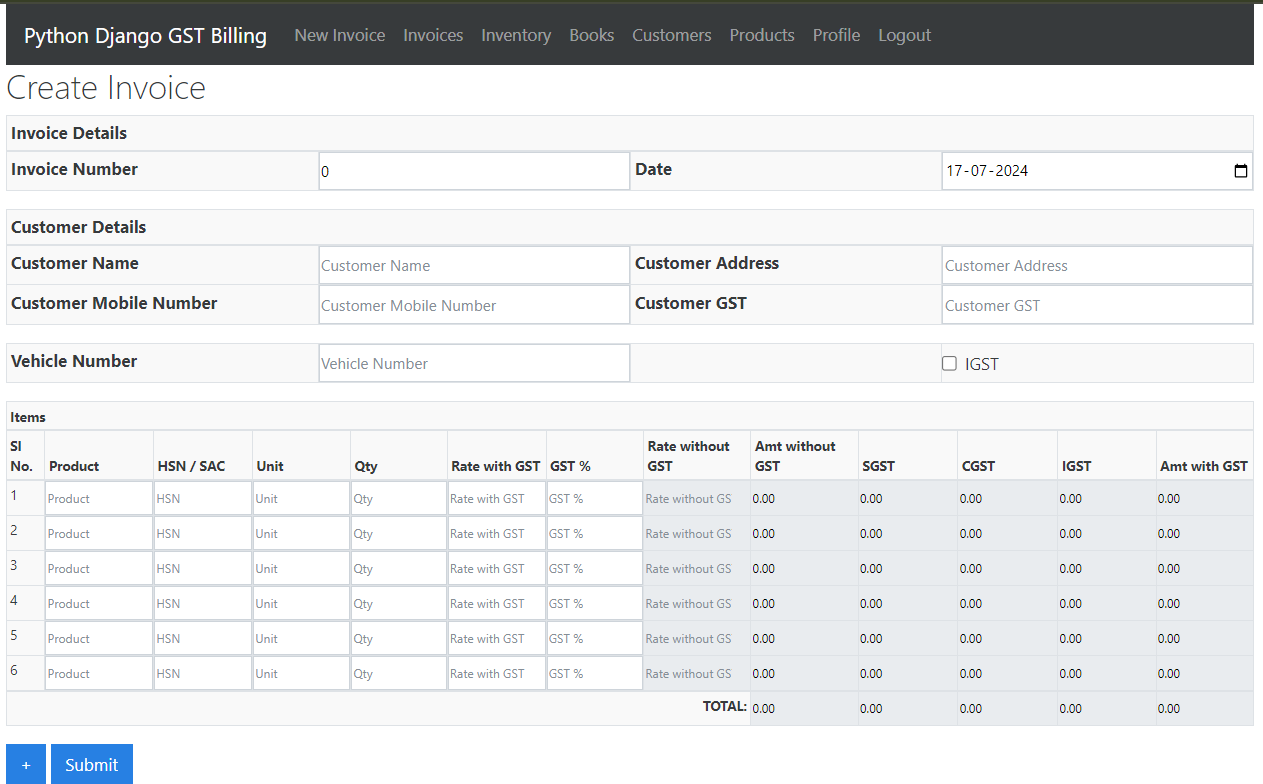


Fig 5.1 Create New invoice

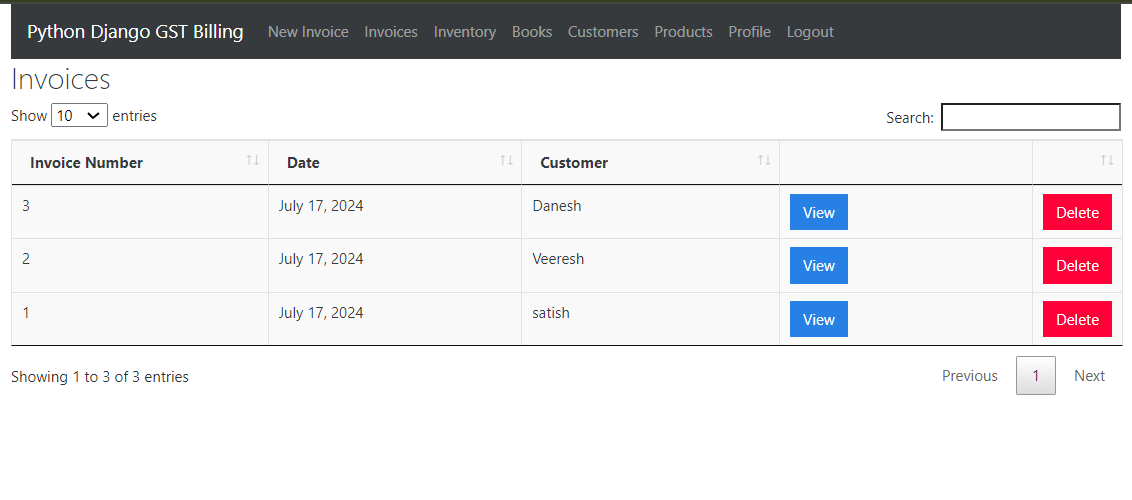


Fig 5.2 Invoices

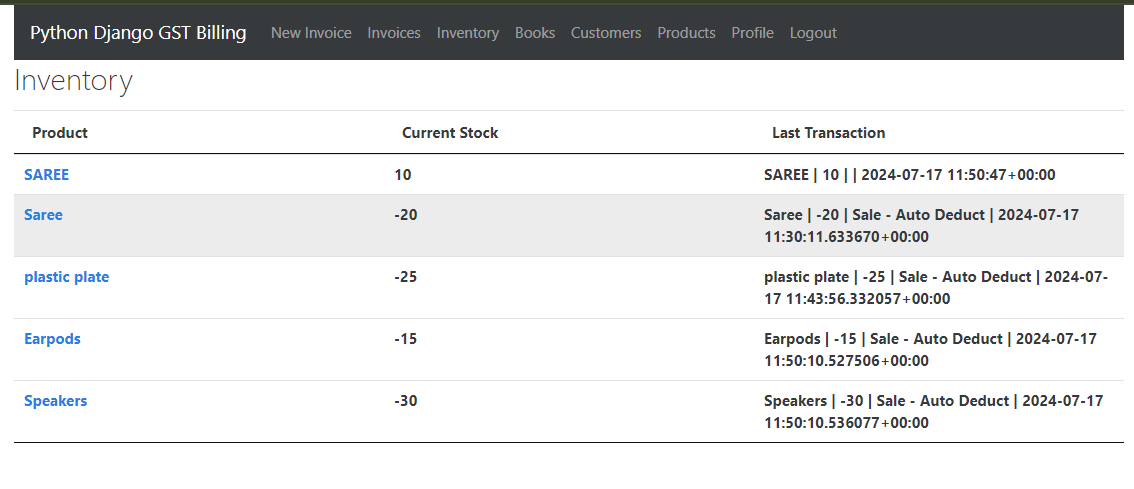


Fig 5.3 Inventory

Fig 5.3 inventory

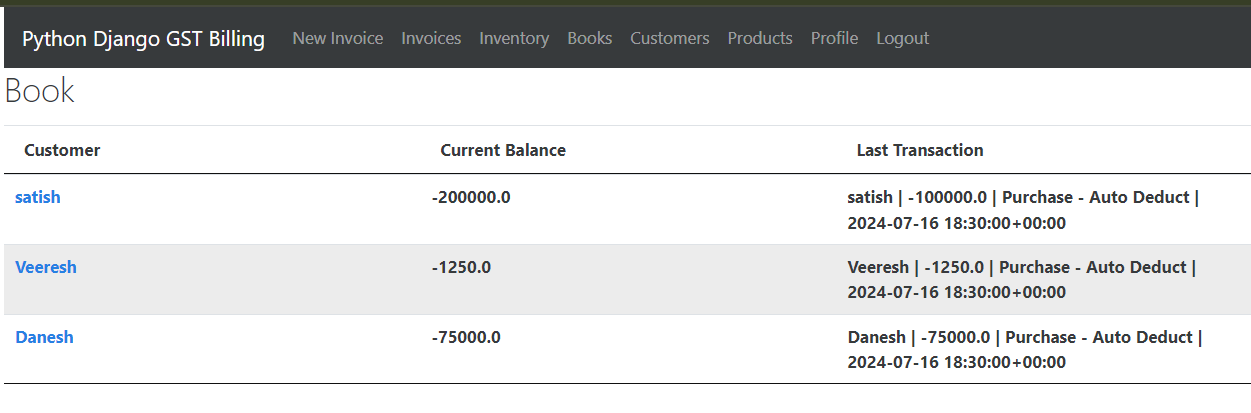
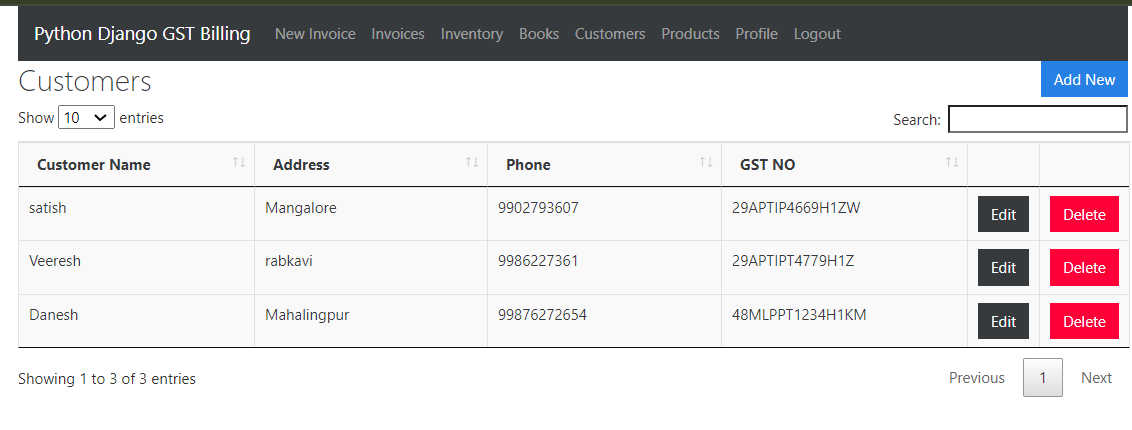


Fig 5.4 Books



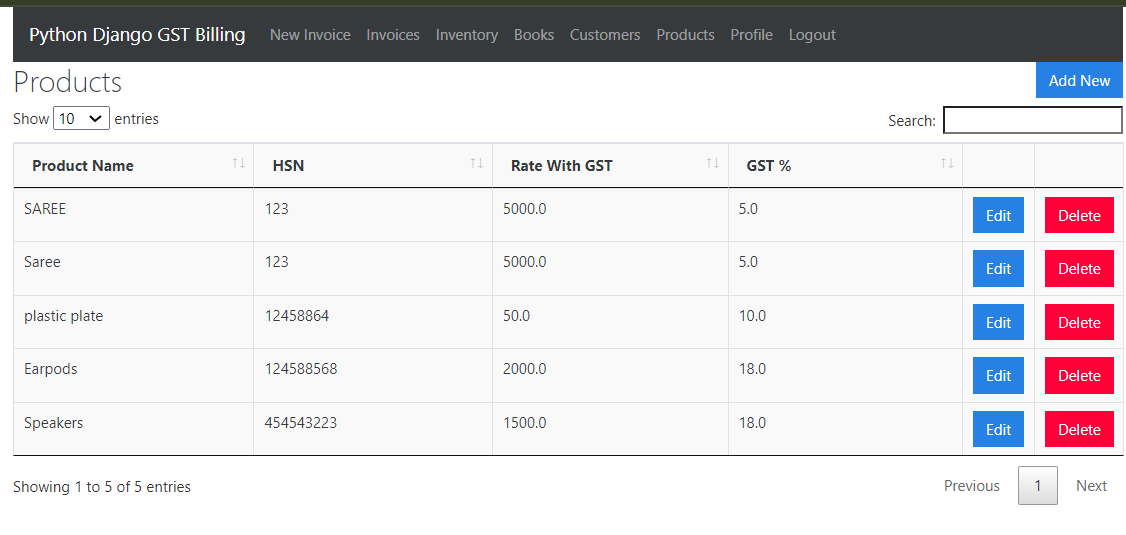
Fig 5.5 Customers

Fig 5.6 Products

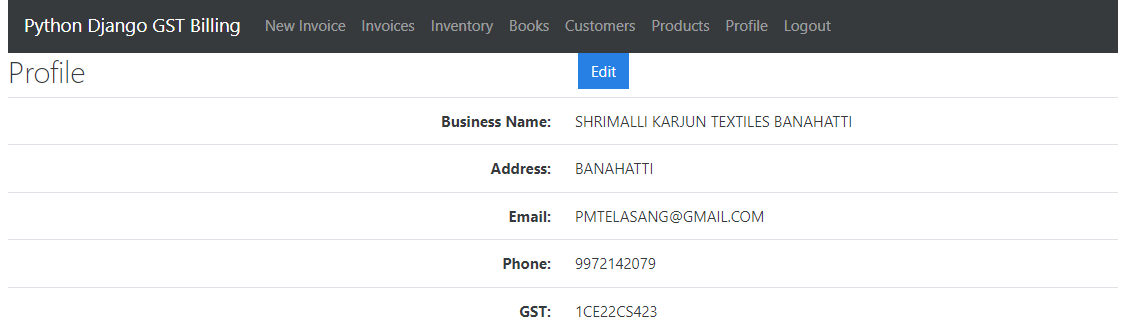


Fig 5.4 Profile Page

**CHAPTER-06**

**CONCLUSION**

In conclusion, we have explored the importance of GST and how a GST Management System can streamline the process of managing GST-related tasks. By automating tasks such as invoice generation, tax calculation, and report generation, businesses can save time, reduce errors, and ensure compliance with tax regulations. The GST Management System not only enhances efficiency but also provides significant cost savings. Implementing such a system is a crucial step for businesses aiming to stay compliant and competitive in the market.

**Feature scope**

The future scope of the GST Management System includes several promising enhancements. Firstly, further automation of manual tasks will streamline GST management processes even more, such as automated reconciliation of invoices and real-time tax calculation updates based on changes in GST rates. Enhanced analytics will be another significant area of development, incorporating advanced analytics to provide deeper insights into financial and tax data. This will include predictive analytics for forecasting tax liabilities and trends, enabling better decision-making with detailed reports and dashboards.

**CHAPTER-07**

**REFERENCES**

**Books:**

1. HTML and CSS: Design and build Websites Publisher: John Wiley and Sons: 1st edition (Nov 8,2011)

2. “Database System Concepts”: Abraham Silberschatz and S Sudarshan

3. “Database Management Systems”: Raghu Ramakrishnan

4. “SQL: QuickStart Guide – The Simplified Beginner’s Guide To SQL” By Clydebank Technology

* www.Tutorialspoint.com
* www.Javatpoint.com
* www.Github.com
* www.Freecodecamp.org
* www.Google.com.
* www.W3schools.com
* www.Getbootstrap.com