

A PROJECT REPORT ON
HOTEL MANAGEMENT SYSTEM

Project report submitted in partial fulfillment of the requirement for the degree of
Bachelor of Technology (Computer Engg.)



Submitted By:
KALPANA PUNDIR
(8721123)

DEPARTMENT OF COMPUTER ENGINEERING
STATE INSTITUTE OF ENGINEERING & TECHNOLOGY, NIlOKHERI
(KURUKSHETRA UNIVERSITY, KURUKSHETRA)
(2023 – 2024)

TABLE OF CONTENTS

i

	Page
Declaration	ii
Certificate	iii
Acknowledgement	iv
Abstract	v
Table of Content	vi
Chapter	
1 Introduction	8
2 Literature Survey	16
3 Methodology Used	18
4 Over-all depth information about project	20
5 Results / Snapshots	34
6 Conclusion and Recommendation	36

DECLARATION

ii

I hereby declare that the Project report entitled " **HOTEL MANAGEMENT SYSTEM** " submitted to the Department of Computer Science & Engineering, SIET Nilokheri in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology (Computer Engineering) is a record of original work done by me, under the guidance and supervision of Sh.----- and it has not formed the basis for the award of any Degree/Diploma/Associateship/ Fellowship or other similar title to any candidate of any University.

Dated

Signature of the Candidate

CERTIFICATE

iii

It is certified that the work contained in the project report titled "HOTEL MANAGEMENT SYSTEM," by "KALPANA," has been carried out under my/our supervision and that this work has not been submitted elsewhere for a degree.

Signature of Supervisor

Name:

Department:

Signature of Head of Department

State Institute of Engg. & Tech. Nilokheri

SB INFOTECH PVT. LTD.
(At Every Step, Innovation is Present)
(An ISO 9001:2015 CERTIFIED COMPANY)
Office Address- 713, New Avas Vikas, Near IV Jain College,
Delhi Road Saharanpur-247001 (U.P)
Phone: +91-7253990102, +91-7252990102
Regd. No. : UP72900UP2022PTC174479


<http://www.sbinfo.in> support@sbinfo.in
Date- 03/09/2023

TO WHOM SO EVER IT MAY CONCERN

This is to certify that Kalpna Pundir Student
successfully completed project "Hotel Management" using
"Python with Django" Technology During REGULAR Training at our
company **SHIVKULBANWARI Infotech Pvt. Ltd.**
Saharanpur.

Batch Date- 21/07/2023 to 03/09/2023

During the training period, he/she was found to be sincere, punctual,
and devoted to his/her Training.


Amit Saini
Managing Director
SHIVKULBANWARI Infotech Pvt. Ltd. Saharanpur- 247001

I am pleased to present the report on “Industrial Training at SB info tech.” I would like to express my sincere thanks to all the people who were in my training process. Their support through the training process was very helpful which helped me to complete my training program successfully. This training has been a pleasurable experience & was able to acquire much knowledge about the hotel industry, due to the help & support extended by my guide. I specially thank Mr. AJEET SAINI

First of all I would like to thank our training Rama for their whole hearted co-operation and I am grateful to my college for facilities extended to us. Lastly I would like to thank all the apprentices of each department with whom I work. They were kind, supportive to me.

A Hotel Management System Project in Django contains different features like, Login and Signup functionality for both manager and customer users, Customers can book room based on availability and See the booking history and edit order from the dashboard.

This project is built on the Django framework and aims to provide a comprehensive solution for managing various aspects of hotel operations.

It consists of several modules that work together seamlessly to streamline processes and enhance efficiency.

The booking management module allows hotels to efficiently handle reservations, including checking availability, managing room types, and assigning rooms to guests.

It simplifies the booking process and ensures smooth operations.

The guest management module helps hotels maintain accurate and up-to-date guest information.

It includes features like guest check-in and check-out, managing guest preferences, and maintaining a guest database for personalized service.

The room management module enables hotels to effectively manage room inventory.

It allows for easy tracking of room availability, maintenance schedules, and room status updates, ensuring optimal utilization of resources.

The billing and invoicing module automates the billing process, generating accurate invoices for guests.

It tracks charges for room bookings, additional services, and any other expenses incurred during the stay.

The reporting and analytics module provides valuable insights into hotel operations.

It generates reports on occupancy rates, revenue, guest preferences, and other key metrics, helping hotels make data-driven decisions.

Overall, the Django hotel management project aims to streamline hotel operations, improve guest experiences, and enhance overall efficiency.

TABLE OF CONTENTS

1. Introduction

- 1.1. Brief Introduction of the project
- 1.2. Objective of the Project
- 1.3. Hardware and software requirements
- 1.4. Brief introduction of the language used

2. Literature Survey

- 2.1. Planning and Requirement
- 2.2. Feasibility Analysis

3. Methodology Used

4. Over-all depth information about project

- 4.1. Depth information about project
- 4.2. Complete code of your project

5. Result and Snapshots

- 5.1. Input and output screen shots with descriptions

6. Conclusion

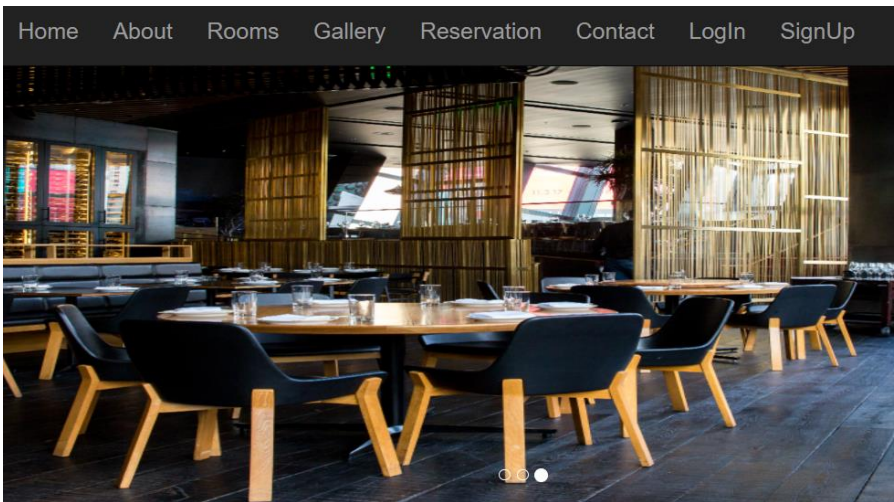
Introduction

1.1. Brief introduction of project

About Project	Project Details	Definition
Project Name	Hotel Management System Project in Django	The Hotel Management System django is developed using Python Django, HTML , CSS and JavaScript , This Hotel Booking App In Django is fully responsive website(for both mobile and larger screens) based on google material design. Also this project has Custom dashboard for both Customer and Room Manager.
Python version (Recommended)	3.8 Version	Python 3.8 introduces some new syntax to the language, as well as a few small modifications to existing behavior and, most importantly, a slew of performance improvements, following in the footsteps of the previous 3.7 version.
Programming Language Used	Python Django Language	Django is a high-level Python web framework for building safe and maintainable websites quickly. Django is a web framework built by experienced developers that takes care of a lot of the heavy lifting so you can focus on developing your app instead of reinventing the wheel.
Developer Name	itsourcecode.com	Free projects containing source code in Java, PHP, Python, Django, VB.Net, Visual Basic, C, C++, C#, Javascript , and other languages are available on this website.
IDE Tool (Recommended)	Visual Studio,	Visual Studio Code, commonly called VS Code, is a versatile, free, open-source code editor by Microsoft. It boasts a robust extension ecosystem, facilitating coding across various languages and frameworks, making it a top choice for developers worldwide.
Project Type	Web Application	A web application, unlike computer-based software programs that operate locally on the device's operating system, is application software that runs on a web server. The user uses a web browser with an active network connection to access web apps.
Database	SQLite	SQLite is a programming language that is used to create embedded software for devices such as televisions, cell phones, and cameras. It can handle HTTP requests with low to medium traffic. SQLite has the ability to compress files into smaller bundles with less metadata. SQLite is a temporary dataset that is used within an application to process data.

1.2. OBJECTIVE OF THE PROJECT

The objective of this project is to design a hotel management systems to manage the details of rooms, customer, booking etc. The main objective is to provide a comprehensive analysis of your Django project for hotel management. This includes discussing the project's purpose, goals, and the specific functionalities it offers. You can delve into the user registration and login system, room booking process, reservation management, check-in/check-out procedures. Additionally, you can highlight the project's impact on streamlining hotel operations, improving efficiency, and enhancing the overall guest experience. Don't forget to mention any challenges faced during development and the strategies you employed to overcome them.



1.3. HARDWARE AND SOFTWARE REQUIREMENTS

For hardware, you'll need the following:

- 1. Computer or Server:** You'll need a computer or server to host your Django project.
- 2. Processor:** Make sure your computer or server has a decent processor to handle the project's workload.
- 3. Memory (RAM):** Aim for at least 4GB of RAM to ensure smooth performance.
- 4. Storage Space:** Ensure you have sufficient storage space to store your project files and any associated data.

For software, you'll need the following:

1. Python: Ensure that you have Python installed on your system. Django is compatible with Python 3.x versions.

Python, created by Guido van Rossum in the late 1980s and first released in 1991, has become one of the most popular and influential programming languages in the world. Renowned for its simplicity, readability, and versatility, Python is used in a vast array of applications across diverse domains.

Python's hallmark is its clean and readable syntax, emphasizing code readability through the use of indentation, reducing the need for braces or semicolons. This readability not only makes it a favorite among developers but also promotes maintainability and collaboration on projects.

Python has a comprehensive standard library that offers modules for various tasks, from file handling and network communication to data manipulation and web development. The language's package manager, pip, simplifies the installation of third-party libraries, contributing to a thriving ecosystem of Python packages.

In web development, Python boasts powerful frameworks like Django and Flask, making it a top choice for building web applications. In data science and artificial intelligence, Python's popularity shines, thanks to libraries such as NumPy, pandas, Matplotlib, and TensorFlow, facilitating tasks like data analysis, visualization, and machine learning.

Python's cross-platform compatibility and support for multiple programming paradigms, including procedural, object-oriented, and functional programming, make it versatile for various applications and industries. Its active community, continuous development, and broad adoption ensure Python's enduring relevance in the ever-evolving world of technology.

2. Django Framework: Django is a powerful and popular open-source web framework written in Python, designed to simplify and streamline web development. Created by Adrian Holovaty and Simon Willison, it was first released in 2005 and has since gained immense popularity for its robust features and rapid development capabilities.

At its core, Django follows the Model-View-Controller (MVC) architectural pattern, but it uses its own interpretation called the Model-View-Template (MVT). This MVT approach separates an application into three main components: Models (representing the data structure), Views (handling the presentation logic), and Templates (defining the HTML layout). This separation of concerns promotes clean, maintainable code and encourages the DRY (Don't Repeat Yourself) principle.

Django offers a plethora of built-in features and tools that accelerate development, including an ORM (Object-Relational Mapping) for database interaction, an admin panel for managing site content, and a URL routing system for handling requests and generating clean, SEO-friendly URLs. Additionally, Django's authentication system, security features, and built-in support for internationalization make it an ideal choice for both small-scale projects and large-scale applications.

Furthermore, Django's extensive ecosystem includes numerous third-party libraries and packages that extend its functionality, allowing developers to add features like user authentication, RESTful APIs, and more with ease. Its robust community, well-maintained documentation, and strong commitment to security make it a reliable choice for web development.

In conclusion, Django is a versatile and mature web framework that empowers developers to build robust, scalable, and maintainable web applications with speed and efficiency.

Its combination of clean code architecture, a rich set of features, and an active community make it an excellent choice for web development projects of all sizes.

3. Database: Choose a database management system that Django supports, such as SQLite and install it on your system. SQLite is a lightweight, serverless, and self-contained relational database management system (RDBMS) that has gained widespread popularity due to its simplicity, versatility, and efficiency. It is a C library that provides a zero-configuration, file-based database engine, making it a popular choice for embedded systems, mobile applications, and smaller-scale web applications.

One of SQLite's key features is its ease of use and minimal setup requirements. It doesn't require a separate server process and operates directly on a single, self-contained file, making it incredibly portable and easy to manage. This simplicity has contributed to its widespread adoption in scenarios where a full-scale database server might be overkill.

Despite its lightweight nature, SQLite is fully ACID-compliant (Atomicity, Consistency, Isolation, Durability), ensuring data integrity and reliability. It supports a wide range of SQL features, including complex queries, transactions, and indexes, making it suitable for a variety of data storage needs.

SQLite's performance is impressive for its size. It excels in read-heavy workloads and can handle moderate write operations efficiently. However, it may not be the best choice for high-concurrency, high-write, or extremely large-scale applications where more robust database systems like MySQL or PostgreSQL are preferred.

SQLite is supported by a multitude of programming languages and platforms, making it highly accessible for developers across different ecosystems. It is often used in mobile app development (both Android and iOS), desktop applications, and embedded systems. Many popular software applications, such as web browsers and operating systems, rely on SQLite for internal data management.

In summary, SQLite is a versatile and widely adopted relational database system known for its lightweight design, ease of use, and compatibility with a wide range of programming languages and platforms. While it may not be suitable for all use cases, its simplicity and performance make it an excellent choice for many applications, particularly those with limited resource requirements.

4. Visual Studio Code: Visual Studio Code (VS Code) is a highly favored code editor for Django development due to its extensive features, excellent Python support, and a wealth of extensions that enhance the development experience.

Python Integration: VS Code offers first-class support for Python, making it a natural choice for Django development. It includes features like code linting, debugging, and autocomplete for Python code, which are indispensable when working on Django projects.

Django Extensions: VS Code provides a range of Django-specific extensions that streamline Django development. These extensions offer support for Django template syntax highlighting, code navigation, and Django management commands integration, simplifying tasks such as creating apps, migrations, and superuser accounts.

Integrated Terminal: VS Code's integrated terminal allows developers to run Django development servers, manage migrations, and execute other project-specific commands without leaving the editor. This seamless integration enhances productivity and keeps the development workflow smooth.

Extensions Marketplace: VS Code boasts a vast extensions marketplace, offering a wide array of Django-related extensions. You can find extensions for Django template snippets, ORM query generation, and even Docker support for Django projects.

1.4. INTRODUCTION OF LANGUAGE

Brief introduction of the language used :- In this project we used the Python language. The brief introduction of the language used show in below:-

Python :- Python is a high-level, general-purpose, and very popular programming language. Python programming language (latest Python 3) is being used in web development, Machine Learning applications, along with all cutting-edge technology in Software Industry.

Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber... etc.

The biggest strength of Python is huge collection of standard library which can be used for the following:

Machine Learning

GUI Applications (like Kivy, Tkinter, PyQt etc.)

Web frameworks like Django (used by YouTube, Instagram, Dropbox)

Image processing (like OpenCV, Pillow)

Web scraping (like Scrapy, BeautifulSoup, Selenium)

Test frameworks

Multimedia

Scientific computing

Text processing and many more..

What can python do ?

In terms of usability,

Python is limitless. It can be used from web development to machine learning, data science and can also be used for writing scripts for automating daily life tasks like sending an email, or desktop notification and many more.

Let's see some of the uses of Python.

How is Python used in Web Development?

Django And Flask are 2 Python-based popular web frameworks that used to create backend code.

Django is an in-demand skill and as it is based on Python the knowledge in Python language enables quick learning of the framework and its application.

Python can be used to build server-side web applications. While a web framework is not required to build web apps, it's rare that developers would not use existing open-source libraries to speed up their progress in getting their application working.

How is Python used in Data Science, Machine Learning?

[Machine Learning](#) And [Data Science](#) have great advantages when used with python this is because of the extensive libraries built in python.

Python is known for using large data and also has its large libraries including [Numpy](#), [Pandas](#) which help in data science and also **TensorFlow** and **Pytorch** that have made matrix computations faster and providing a boost in the Data Science world.

How is Python used in Desktop Application Development?

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit. Python provides various options for developing graphical user interfaces (GUIs). Most important are listed below.

Django and its Installation

Introduction to Django

Django is a high-level, open-source web framework written in Python that follows the Model-View-Controller (MVC) architectural pattern, although Django uses its own interpretation called the Model-View-Template (MVT). It is designed to simplify and expedite web development by providing a robust set of tools and conventions for common tasks, allowing developers to focus on creating web applications quickly and efficiently.


Django was created by Adrian Holovaty and Simon Willison and was first released in 2005. Since then, it has gained immense popularity and has been used to build a wide range of web applications, from simple blogs to complex e-commerce platforms and social media sites. Some of the notable websites built with Django include Instagram, Pinterest, and The Washington Times.

Django is a popular Python web framework that simplifies the development of web applications. With Django, you can build robust and scalable web applications with ease. One of the key features of Django is its powerful ORM (Object-Relational Mapping) system, which allows you to interact with databases using Python code instead of writing SQL queries. This makes it easier to work with data and ensures that your application is secure against common security vulnerabilities like SQL injection. Django also provides a built-in admin interface, which allows you to easily manage your application's data and perform CRUD (Create, Read, Update, Delete) operations without writing any code. This is particularly useful for quickly prototyping and managing the backend of your application. Additionally, Django has a strong and supportive community, with a vast ecosystem of third-party packages and libraries that extend its functionality. This means that you can easily find solutions to common problems and leverage existing code to speed up development. Overall, Django is a fantastic choice for building web applications. Its simplicity, scalability, and emphasis on best practices make it a favorite among

Installation of Django

Installing Django is a straightforward process, thanks to its dedicated package manager and the Python ecosystem. Before you begin, ensure that you have Python installed on your system. You can check your Python version by opening a terminal or command prompt and running:

shell

 Copy code

```
python --version
```

To install Django, follow these steps:

Step 1: Create a Virtual Environment (Optional but Recommended)

Creating a virtual environment is considered best practice to isolate your project's dependencies from the global Python environment. It helps avoid conflicts and ensures that your project uses the specific versions of libraries you need. To create a virtual environment, run the following command:

```
Shell for new features  
> pip install pipenv
```

Step 2: Activate the Virtual Environment

Activate the virtual environment by running the appropriate command based on your operating system:

```
> pipenv shell
```

Step 3: Install Django

With your virtual environment activated, you can now use pip, the Python package manager, to install Django. Run the following command:

```
> pip install django
```

This command will download and install the latest version of Django into your virtual environment.

Step 4: Verify the Installation

After the installation is complete, you can verify that Django has been installed successfully by running the following command:

```
> py manage.py runserver
```

This should confirm that Django is now available in your virtual environment.

2. Literature Survey

2.1. Planning and Requirement

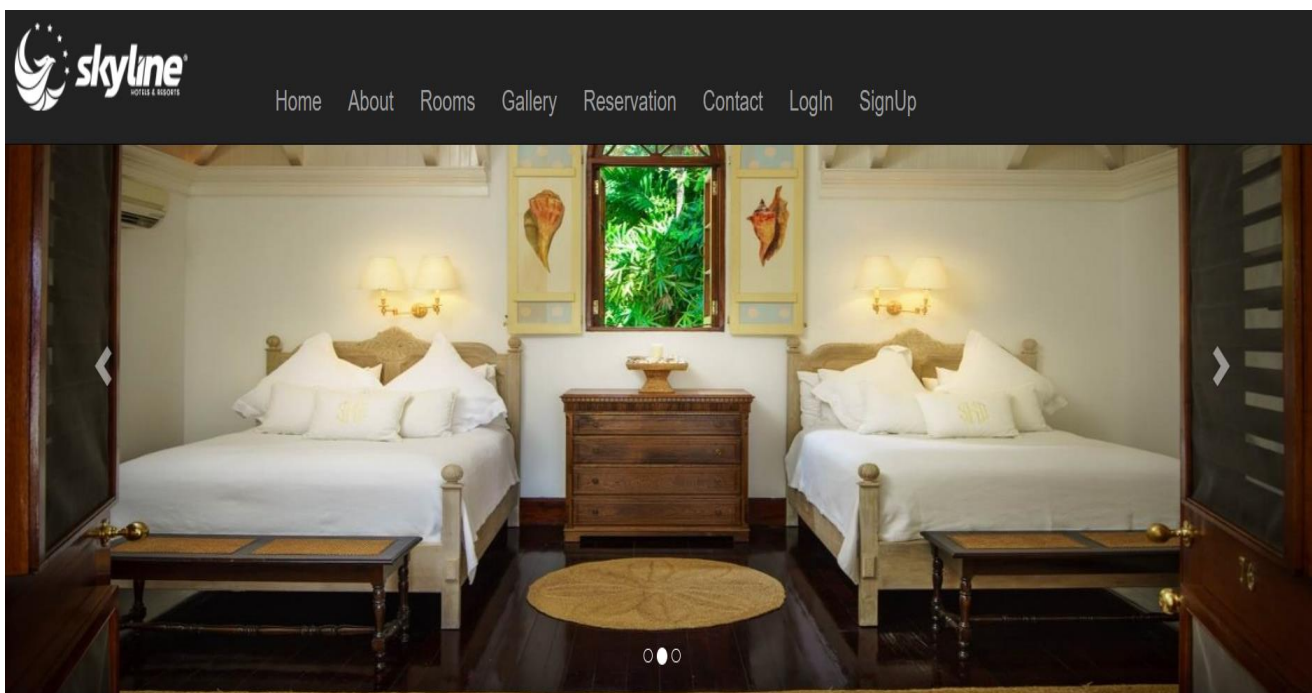
We'll start by identifying the project's requirements, such as managing hotel bookings, room availability, and guest information.

Next, we'll assess the system's architecture, including components like the database, user interface, and booking management system.

We'll also consider the project's performance, reliability, and scalability to ensure it can handle a large number of bookings and users.

Additionally, we'll evaluate the project's usability, making sure it's intuitive for hotel staff to use.

Overall, a system analysis of this Django hotel management project involves assessing its requirements, architecture, performance, and usability.



2.2. FEASIBILITY ANALYSIS

MARKET ANALYSIS:-

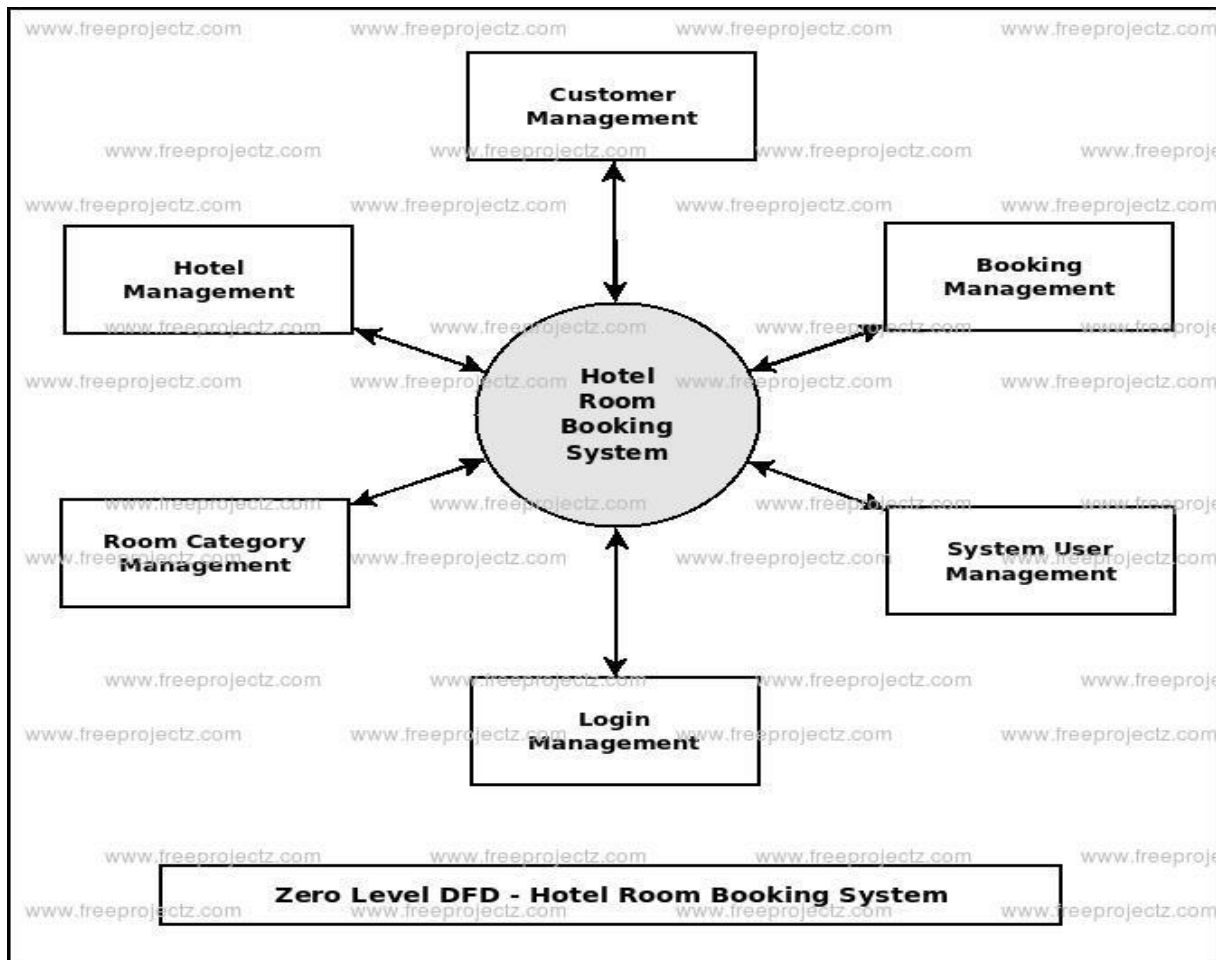
By analyzing the market, we can understand the demand for such a system in the hospitality industry.

The hotel management software market is growing rapidly, with a need for efficient and automated solutions.

Django's flexibility and scalability make it a popular choice for building hotel management systems.

With features like booking management, room availability tracking, and guest information management, the Django hotel management project has the potential to meet the needs of hotels and resorts.

However, it's important to consider competition and market trends to ensure the project stands out.



TECHNOLOGY ANALYSIS:-

In this project we require many technology and many package. They are shown in below:

- Python
- Django
- Html
- Css
- Javascript

3. METHODOLOGY USED

A Hotel Management System Project in Django contains different features like, Login and Signup functionality for both manager and customer users, Customers can book room based on availability and See the booking history and edit order from the dashboard.

Following Method Used:

Certainly! Let's break down the methodology into theoretical steps:

1. Project Setup:

- Establish the foundation of your project by creating a new Django project using the ``django-admin`` command. This project will serve as the container for your entire application.

2.App Creation:

- Divide your project into modular components called "apps." For a hotel management system, create a Django app specifically for handling hotel-related functionalities.

3. Define Models:

- Identify the essential entities in your hotel management system, such as ``Hotel``, ``Room``, and ``Reservation``. Define these entities as models in Django, specifying their attributes and relationships.

4. Run Migrations:

- Utilize Django's migration system to convert your model definitions into database tables. This involves creating migration files and applying them to create the necessary database schema.

5. Create Views:

- Develop views to handle user requests and interact with the data models. Views encapsulate the logic that retrieves data from the database and renders it for presentation.

6.Create Templates:

- Design HTML templates to structure the user interface and display the information retrieved by views. Templates play a crucial role in separating the presentation layer from the application logic.

7. Configure URLs:

- Establish URL patterns that map to specific views. This ensures that when a user accesses a particular URL, the corresponding view is invoked to handle the request.

8. Integrate Templates with Views:

- Connect the HTML templates with Django views, allowing for dynamic content rendering. Views pass data to templates, enabling the creation of dynamic web pages.

9. User Authentication:

- Implement user authentication mechanisms if needed. Django provides built-in functionality for user authentication, including user registration, login, and logout.

10. Static Files and Media:

- Manage static files (e.g., CSS, JavaScript) and media files (e.g., images, file uploads) to enhance the visual appeal and functionality of your application.

11. Testing:

- Perform thorough testing of your application to identify and rectify any issues. This includes unit tests, integration tests, and end-to-end testing to ensure the application behaves as expected.

12. Deployment:

- Prepare your application for deployment to a production environment. Choose a suitable hosting platform, configure server settings, and deploy the application to make it accessible to users.

13. Monitoring and Maintenance:

- Implement monitoring tools to keep track of the application's performance and user interactions. Regularly update the application to incorporate new features, fix bugs, and address security concerns.

This theoretical methodology provides a structured approach to building a hotel management system using Django in Python. The process involves clear steps from project initiation to deployment, with a focus on modular development, user interface design, and system reliability.

4. Over-all depth information about project

4.1. Depth information:

Certainly! Let's delve into a theoretical overview of a hotel management project using Django in Python.

1. Project Setup:

- Description:

- Create a new Django project using the ``django-admin`` command.

- Steps:

- Execute ``django-admin startproject hotel_management``.
- Navigate into the project directory using ``cd hotel_management``.

2. App Creation:

- Description:

- Divide the project into modular apps to handle specific functionalities.

- Steps:

- Create a Django app for hotels: ``python manage.py startapp hotels``.
- Repeat for additional functionalities (rooms, reservations, users).

3. Models:

- Description:

- Define data models to represent entities such as hotels, rooms, and reservations.

- Example:

- ``Hotel`` model with attributes like name, location, and facilities.
- ``Room`` model with attributes like room number, type, price, and availability.
- ``Reservation`` model with attributes like guest information, check-in, and check-out dates.

4. Run Migrations:

- Description:

- Use Django migrations to create database tables based on the defined models.

- Steps:

- Run ``python manage.py makemigrations``.
- Run ``python manage.py migrate``.

5. User Authentication:

- Description:

- Implement user authentication for secure access to reservation and booking features.

- Steps:

- Utilize Django's built-in authentication views and models.
- Enable user registration, login, and logout.

6. Views and Templates:

- Description:

- Develop views to handle user requests and render data using HTML templates.

- Example:

- ``hotel_list`` view to display a list of hotels.
- ``room_details`` view to show information about a specific room.

7. URL Configuration:

- Description:

- Configure URL patterns to map to specific views within each app.

8. Forms:

- Description:

- Use Django forms to handle user input for reservations, user registration, etc.

- Steps:

- Create a reservation form to gather guest information and booking details.

9. User Interface (Templates):

- Description:

- Design HTML templates to structure the user interface and present dynamic content.

- Example:

- Create templates for hotel lists, room details, and reservation forms.

10. Static Files and Media:

- Description:

- Manage static files (CSS, JS) for styling and interactivity.
- Handle media files (images) for hotel and room images.

- Steps:

- Configure static and media file settings in Django.

11. Middleware:

- Description:

- Use Django middleware for additional functionalities.

- Example:

- Implement middleware for tracking user activity or managing sessions.

12. Django Admin Panel:

- Description:

- Leverage the built-in Django admin panel for easy management of database records.

- Steps:

- Register models with the admin panel to allow admins to add, edit, and remove data.

13. Testing:

- Description:

- Develop and run tests to ensure the correct behavior of models, views, and forms.

- Steps:

- Create unit tests and integration tests for key functionalities.

14. Security:

- Description:

- Implement security measures to protect against common web vulnerabilities.

- Steps:

- Ensure secure data storage, validate user input, and handle authentication securely.

15. Deployment:

- Description:

- Deploy the application to a web server (e.g., Heroku, AWS, DigitalOcean).

- Steps:

- Configure production settings, set up databases, and deploy the application.

4.2. CODING

1. **SETTINGS.PY :**

#source code

"""

Django settings for djangoproj project.

Generated by 'django-admin startproject' using Django 4.2.4.

For more information on this file, see

<https://docs.djangoproject.com/en/4.2/topics/settings/>

For the full list of settings and their values, see

<https://docs.djangoproject.com/en/4.2/ref/settings/>

"""

from pathlib import Path

Build paths inside the project like this: BASE_DIR / 'subdir'.

BASE_DIR = Path(__file__).resolve().parent.parent

Quick-start development settings - unsuitable for production

See <https://docs.djangoproject.com/en/4.2/howto/deployment/checklist/>

SECURITY WARNING: keep the secret key used in production secret!

SECRET_KEY = 'django-insecure-#pl%q0^+40*y#)^c=9iw(%3qb^^!*5i4fa&s63%cl#4%7sre+'

SECURITY WARNING: don't run with debug turned on in production!

DEBUG = True

ALLOWED_HOSTS = [

Application definition

INSTALLED_APPS = [

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

'myapp',

]

EMAIL_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'

EMAIL_HOST = 'smtp.gmail.com'

EMAIL_USE_TLS = True

EMAIL_PORT = 587

EMAIL_HOST_USER = 'kalpanapundir3@gmail.com'

EMAIL_HOST_PASSWORD = "rwzpfvtvegtzcmak"

MIDDLEWARE = [

'django.middleware.security.SecurityMiddleware',

'django.contrib.sessions.middleware.SessionMiddleware',

'django.middleware.common.CommonMiddleware',

'django.middleware.csrf.CsrfViewMiddleware',

'django.contrib.auth.middleware.AuthenticationMiddleware',

'django.contrib.messages.middleware.MessageMiddleware',

```

'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'djangoproj.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    ],
]
# Default primary key field type
# https://docs.djangoproject.com/en/4.2/ref/settings/#default-auto-field

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'

```

2. **URLS.PY :**

#source code

"""

URL configuration for djangoproj project.

The `urlpatterns` list routes URLs to views. For more information please see:

<https://docs.djangoproject.com/en/4.2/topics/http/urls/>

Examples:

Function views

1. Add an import: from my_app import views
2. Add a URL to urlpatterns: path("", views.home, name='home')

Class-based views

1. Add an import: from other_app.views import Home
2. Add a URL to urlpatterns: path("", Home.as_view(), name='home')

Including another URLconf

1. Import the include() function: from django.urls import include, path
2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))

"""

```

from django.contrib import admin
from django.urls import include,path

```

```

urlpatterns = [
    path('admin/', admin.site.urls),
    path("",include('myapp.urls')),
]

```

TEMPLATES CODE

3. BASE.HTML:

#source code

```
{% load static %}

<html>

<head>

<!-- Latest compiled and minified CSS -->

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.0/css/bootstrap.min.css"

<!-- jQuery library -->

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>

<!-- Late
st compiled JavaScript -->

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.0/js/bootstrap.min.js"></script>

<link href="{% static 'css/style-index.css' %}" rel="stylesheet" type="text/css"

</head>

<body>

<!--navbar-->

<nav class="navbar navbar-inverse navbar-fixed-top">

  <div class="container-fluid">

    <div class="navbar-header">

      <a class="col-lg-3 col-xs-12 navbar-brand" href="#"></a>

</a>

    </div>

    <ul class="nav navbar-nav">

      <li><a href="index">Home</a></li>

      <li><a href="about">About</a></li>

      <li><a href="room">Rooms</a></li>

      <li><a href="gallery">Gallery</a></li>

      <li><a href="reservation">Reservation</a></li>

      <li><a href="contact">Contact</a></li>

      <li><a href="/">LogIn</a></li>

      <li><a href="signup">SignUp</a></li>

    </ul>
```



```

</div>

</nav>

<!--navbar end-->

{% block content %}

{% endblock content %}

<!--Footer-->

<div class="col-lg-12 footer">

    <div class="container">

<!--part 1 logo-->

        <div class="col-lg-3 col-xs-12 part1">

            <center>

            </center>

        </div>

<!--Footer ends-->

</body>

</html>

```

4. ABOUT.HTML:

#source code

```

"""
{% extends "base.html" %}
{% load static %}
{% block content %}

<html>
<head>

    <!-- Latest compiled and minified CSS -->
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.0/css/bootstrap.min.css">

    <!-- jQuery library -->
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>

    <!-- Late
st compiled JavaScript -->
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.0/js/bootstrap.min.js"></script>

    <link href="{% static 'css/about.css' %}" rel="stylesheet" type="text/css">
</head>

<body>

```

```

```

```
<div class="col-lg-12 col-xs-12 block-2">
```

```
  <div class="container">
```

```
    <div class="col-lg-6 col-xs-12 left-part">
```

```
      
```

```
    </div>
```

```
    <div class="col-lg-6 col-xs-12 right-part">
```

```
      <h1 class="heading">About us</h1>
```

```
      <p>
```

It is a long established fact that a reader will be distracted by
the readable content of a page when looking at
its layout. The point of using Lorem Ipsum is that it has a more-
or-less normal distribution of letters, as opposed

to using 'Content here, content here', making it look like
readable English. Many desktop publishing packages and
web page editors now use Lorem Ipsum as their default model
text, and a search for 'lorem ipsum' will uncover many
web sites still in their infancy.

There are many variations of passages of Lorem Ipsum
available, but the majority have suffered alteration in some
form, by injected humour, or randomised words which don't
look even slightly believable. If you are going to use
a passage of Lorem Ipsum, you need to be sure
</p>

```
    </div>
```

```
  <hr class="line">
```

```
  <div class="col-lg-6 col-xs-12 left-bottom">
```

```
    <h1 class="heading">Why Guest Choose Skyline Hotel?</h1>
```

```
    <p>
```

It is a long established fact that a reader will be distracted by
the readable content of a page when looking at its layout.

The point of using Lorem Ipsum is that it has a more-or-less
normal distribution of letters, as opposed to using 'Content here,
content here', making it look like readable English. Many
desktop publishing packages and web page editors now use Lorem Ipsum
as their default model text, and a search for 'lorem ipsum' will
uncover many web sites still in their infancy. Various versions
have evolved over the years, sometimes by accident, sometimes
on purpose (injected humour and the like).

```
    </p>
```

```
  </div>
```

```
  <div class="col-lg-6 col-xs-12 right-bottom">
```

```
    
```

```

        </div>
    </div>
</div>
{% endblock content %}

```

5. CONTACT.HTML:

#source code

```

{% extends "base.html" %}
{% load static %}
{% block content %}
<html>
<head>
    <!-- Latest compiled and minified CSS -->
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.0/css/bootstrap.min.css">

    <!-- jQuery library -->
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>

    <!-- Late
st compiled JavaScript -->
    <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.0/js/bootstrap.min.js"></script>

    <link href="{% static 'css/contact.css' %}" rel="stylesheet" type="text/css">
<link href="https://stackpath.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet" type="text/css">

</head>
<body>

<!--Contact form-->
<div class="container">
    <div class="col-lg-12 col-xs-12 contact">
<!--left part-->
        <div class="col-lg-6 col-xs-12 details">
            <h1 class="heading">Contact Us</h1>
            <p>Send in your enquiries and we will reply to you as soon as
possible.

            Alternatively, you may contact our City offices for further
information.

            </p>

            <ul><i class="fa fa-map-marker" aria-
hidden="true"></i></i> Address-121 King Str, Melbourne Victoria</ul>
            <ul><i class="fa fa-phone-square" aria-
hidden="true"></i> Mobile-1-548-854-8898</ul>

```

```

        <ul><i class="fa fa-envelope" aria-hidden="true"></i>
E-mail-skylinehotel@gmail.com</ul>

    </div>

<!--left part ends-->

<!--right part-->
    <div class="col-lg-6 col-xs-12 form">
        <h1 class="heading">Quick Inquiries</h1>
        <form class="contact-form" action="contactForm"
method="post">{% csrf_token %}
            <input class=" text" type="text" placeholder="Name"
name="name">
            <input class=" sub"type="text" placeholder="Mobile
no." name="phn">
            <input class=" text" type="email" placeholder="E-
Mail" name="email">
            <input class=" sub"type="text" placeholder="subject"
name="subject">
            <input class=" msg" type="text" placeholder="Write
what do you want" name="msgs">

            <button type="submit" style="margin-top: 10px;
margin-left: 230px; background-color: rebeccapurple;color: white;">Submit</button>

        </form>

    </div>
</div>

{% endblock content %}

```

6. GALLERY.HTML:

#source code

```

{% extends "base.html" %}
{% load static %}
{% block content %}
<html>
<head>
    <!-- Latest compiled and minified CSS -->
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.0/css/bootstrap.min.css">

    <!-- jQuery library -->
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>

```

```

        <!-- Latest compiled JavaScript -->
        <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.0/js/bootstrap.min.js"></script>

        <link href="{% static 'css/gallery.css' %}" rel="stylesheet" type="text/css">
</head>
<body>
<!--gallery-->
<div class="container">

<h1>Gallery</h1>
    <div class="col-lg-12 col-xs-12 block-3">

        <h1 class="heading">Gallery</h1><br><br>
        <div class="col-lg-4 col-xs-12 imgs">
            <a href="{% static 'images/our-3.jpg' %}"></a>
        </div>
        <div class="col-lg-4 col-xs-12 imgs">
            <a href="{% static 'images/our-4.jpg' %}"></a>

</div>

<!--gallery !-->

{% endblock content %}

</body>
</html>

```

7. INDEX.HTML:

#source code

```

{% extends "base.html" %}

{% load static %}

{% block content %}

```

```

<!--slider -->

```

```

<div class=" slider">

```

```

<div id="myCarousel" class="carousel slide" data-ride="carousel">

```

<!-- Indicators -->

<ol class="carousel-indicators">

<li data-target="#myCarousel" data-slide-to="0" class="active">

<li data-target="#myCarousel" data-slide-to="1">

<li data-target="#myCarousel" data-slide-to="2">

<!-- Wrapper for slides -->

<div class="carousel-inner">

<div class="item active">

</div>

<div class="item">

</div>

<div class="item">

</div>

<div class="item">

</div>

</div>

<div class="col-lg-4 col-xs-12 img">

</div>

</center>

</div>

<div class="col-lg-12 col-xs-12 row-2">

<center>

View More

</center>

</div>

</div>

<!--gallery !-->

{% endblock content %}

ADMIN.PY

Source code

```
from django.contrib import admin
```

```
from .models import Contact
```

```
# Register your models here.
```

```
admin.site.register(Contact)
```

MODELS.PY

Source code

```
from django.db import models
```

```
# Create your models here.
```

```
class Contact(models.Model):
```

```
    Name=models.CharField(max_length=50)
```

```
    Phone=models.IntegerField()
```

```
    Email=models.EmailField()
```

```
    Subject=models.TextField()
```

```
    Message=models.TextField()
```

URLS.PY

```
#Source code of urls.py in myapp [application]
```

```
from django.urls import path
```

```
from . import views
```

```
urlpatterns=[
```

```
    path("",views.login,name='login'),
```

```
    path('index',views.index,name='index'),
```

```
    path('about',views.about,name='about'),
```

```
    path('contact',views.contact,name='contact'),
```

```
    path('gallery',views.gallery,name='gallery'),
```

```
    path('reservation',views.reservation,name='reservation'),
```

```
    path('room',views.room,name='room'),
```

```
    path('signup',views.signup,name='signup'),
```

```

    path('contactForm',views.contactForm,name='contactForm'),
    path('reservationForm',views.reservationForm,name='reservationForm'),
    path('loginForm',views.loginForm,name='loginForm'),
    path('signupForm',views.signupForm,name='signupForm'),
]

```

VIEWS.PY

Source code

```

from django.shortcuts import render
from django.http import HttpResponse
from .models import Contact
from django.conf import settings
from django.core.mail import send_mail
from django.contrib.auth.models import User
from django.contrib.auth import authenticate,login

```

```

# Create your views here.

```

```

def index(request):
    return render(request,"index.html",{})

```

```

def about(request):
    return render(request,"about.html",{})

```

```

def contact(request):
    return render(request,"contact.html",{})

```

```

def gallery(request):
    return render(request,"gallery.html",{})

```

```

def reservation(request):
    return render(request,"reservation.html",{})

```

```

def room(request):
    return render(request,"room.html",{})

```

```

def login(request):
    return render(request,"login.html",{})

```

```

def signup(request):
    return render(request,"signup.html",{})

```

```

def contactForm(request):
    name1=request.POST.get('name','default')
    print(name1
    else:

```



```
user=User.objects.create(username=username,email=email1,password=password1)
user.set_password(password1)
user.save()
return render(request,'login.html',{'})
```

BRIEF DESCRIPTION OF MODULE DEVELOPED:

The Django hotel management project consists of various modules that work together to streamline hotel operations. These modules include booking management, guest management, room management, billing and invoicing, and reporting and analytics. Each module has its own set of functionalities to handle different aspects of hotel management.

HttpResponse (): HttpResponse (source code) provides an inbound HTTP request to a Django web application with a text response. This class is most frequently used as a return object from a Django view.

Settings: A Django settings file contains all the configuration of your Django installation. This document explains how settings work and which settings are available.

Send_mail: Although Python provides a mail sending interface via the **smtplib** module, Django provides a couple of light wrappers over it. These wrappers are provided to make sending email extra quick, to help test email sending during development, and to provide support for platforms that can't use SMTP.

auth.models: This document explains the usage of Django's authentication system in its default configuration. This configuration has evolved to serve the most common project needs, handling a reasonably wide range of tasks, and has a careful implementation of passwords and permissions. For projects where authentication needs differ from the default, Django supports extensive extension and customization of authentication.

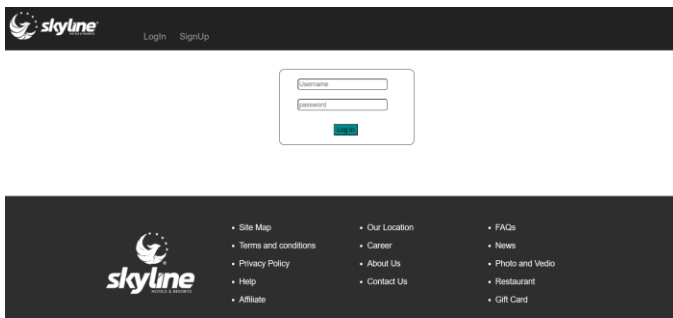
Django authentication provides both authentication and authorization together and is generally referred to as the authentication system, as these features are somewhat coupled.

User objects

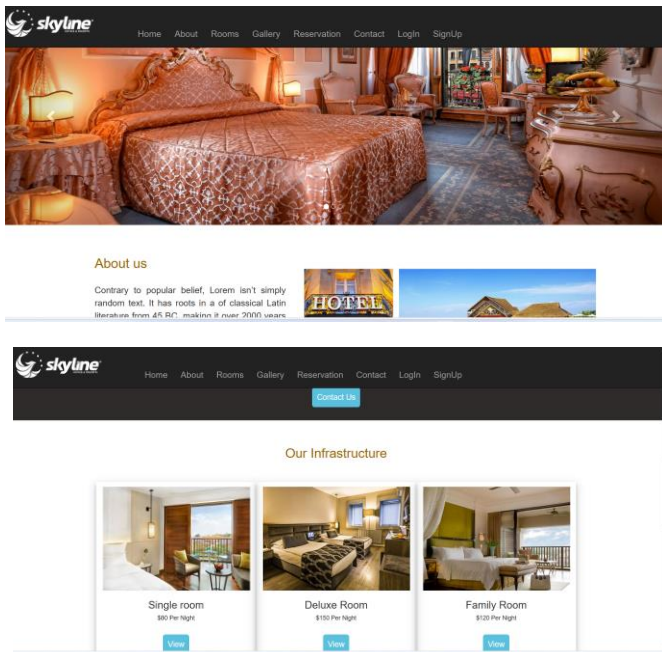
User objects are the core of the authentication system. They typically represent the people interacting with your site and are used to enable things like restricting access, registering user profiles, associating content with creators etc. Only one class of user exists in Django's authentication framework, i.e., '**superusers**' or admin '**staff**' users are just user objects with special attributes set, not different classes of user objects.

5. Results and Snapshots:

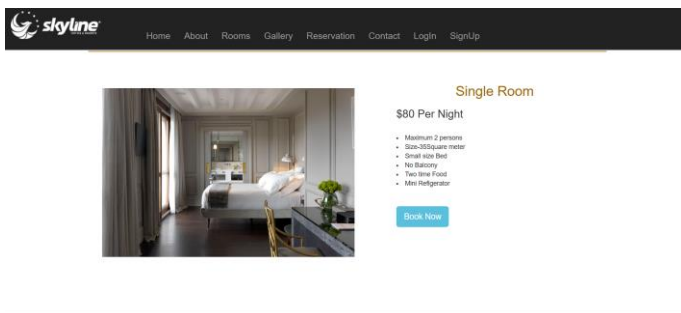
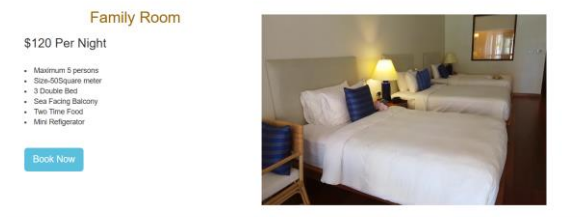
Login page:



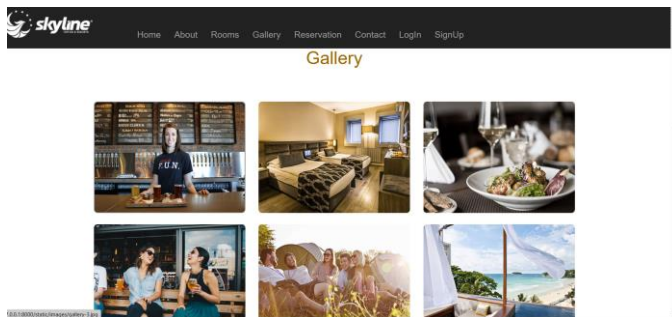
Home page:



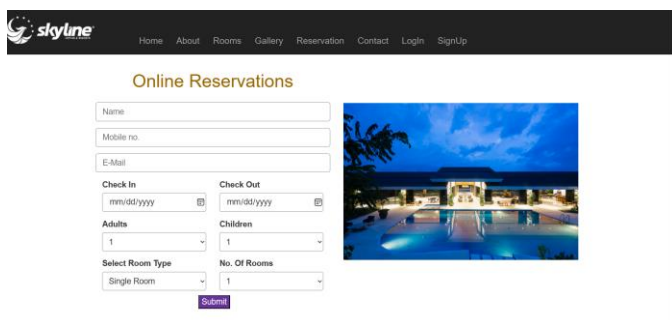
Room page:



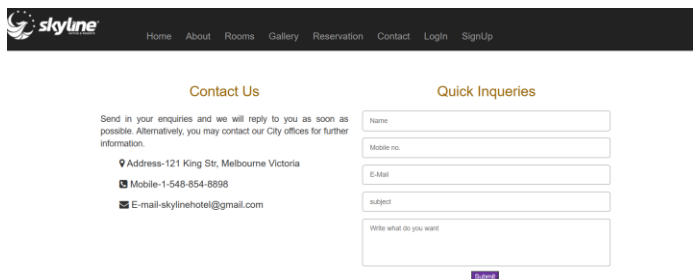
Gallery page:



Reservation page:



Contact page:



6. Conclusion

In conclusion, the Django hotel management project is a robust and comprehensive solution for hotels to efficiently manage their operations. By leveraging the power of the Django framework, this project offers a range of modules that work seamlessly together to streamline processes and enhance overall efficiency.

With features like booking management, guest management, room management, billing and invoicing, and reporting and analytics, hotels can effectively handle reservations, maintain accurate guest information, manage room availability, generate accurate invoices, and gain valuable insights into their operations.

The booking management module simplifies the reservation process, ensuring that rooms are allocated efficiently and accurately. The guest management module helps hotels maintain personalized guest profiles and provide tailored services. The room management module allows for easy tracking of room availability and maintenance schedules. The billing and invoicing module automates the billing process, reducing manual errors and generating accurate invoices. The reporting and analytics module provides valuable insights into occupancy rates, revenue, and guest preferences, enabling data-driven decision-making.

By implementing the Django hotel management project, hotels can streamline their operations, improve guest experiences, and optimize resource utilization. This project offers a user-friendly interface, scalability, and flexibility, making it suitable for hotels of all sizes.

Overall, the Django hotel management project is a comprehensive and efficient solution that empowers hotels to effectively manage their operations, enhance guest satisfaction, and drive business growth.