Project Title: Online Cruise Booking System

(By SEMESTER – 9th of IV Year M.Sc. (2025-2))

Submitted By:

Student Name: Roll no.

1). Gajjar Helly Pinankkumar, 5021

2). Mistry Swati Hemantkumar, 5040

Date of submission: 15/12/2025

Submitted To

K. S. School of Business Management &amp; Information Technology

M.Sc. - Computer Applications and Information Technology.

CHAPTER – 1

INTRODUCTION

1.1 ORGANIZATION PROFILE

The organization behind the development of the Online Cruise Booking System is a software development and training company that works in the field of web technologies and full stack development. The organization mainly focuses on developing web-based applications using modern technologies such as Python, Django, HTML, CSS, JavaScript, and MySQL. Along with development services, the organization also provides internship training and project guidance to students.

The organization follows a professional software development life cycle to design, develop, test, and deploy applications. It believes in delivering user-friendly, secure, and scalable software solutions that solve real-world problems. The development team consists of project managers, senior developers, junior developers, testers, and interns who work together to complete projects efficiently.

The organization encourages practical learning by allowing interns to work on live projects. During the development of the Online Cruise Booking System, the organization provided proper guidance, technical support, and resources. The project was developed by following industry standards and best practices. Regular reviews and testing were conducted to ensure quality and performance.

The organization uses open-source tools and frameworks to reduce development cost while maintaining high quality. It also focuses on customer satisfaction by understanding user requirements clearly and implementing them effectively. Overall, the organization plays an important role in providing a professional environment for developing reliable and efficient software systems like the Online Cruise Booking System.

1.2 SYSTEM DETAILS

The Online Cruise Booking System is a web-based application designed to simplify the process of booking cruise trips. The system allows users to search for available cruises, nearby hotels, and onboard activities through a single platform. It replaces the traditional manual booking process with an automated and digital solution.

The system is developed using full stack technologies where the frontend is designed using HTML, CSS, and JavaScript to provide an interactive user interface, while the backend is developed using Python Django to manage business logic, database operations, and security. MySQL is used as the database to store user information, booking details, and payment records.

The system supports user registration and login, cruise search, booking management, online payment, and email notifications. It also includes an admin panel that allows administrators to manage cruises, hotels, users, and bookings efficiently. The system is designed to be secure, fast, and easy to use for both users and administrators.

1.2.1 EXISTING SYSTEM

In the existing system, cruise booking was mostly done through travel agents or multiple separate online platforms. Users had to visit different websites for cruise booking, hotel booking, and activity planning. This made the entire process complicated and time-consuming. In many cases, users had to depend on agents for information, which increased the cost and reduced transparency.

The existing system lacked real-time updates about cruise availability, pricing, and offers. Manual paperwork and phone-based communication were common, which often resulted in errors, delays, and confusion. Payment processes were also not fully secure or convenient in many cases.

There was no centralized system to manage all booking-related information. Users could not easily track their bookings or receive instant confirmations. Admin management was also difficult due to the absence of proper digital records. Overall, the existing system was inefficient, slow, and user-unfriendly.

1.2.2 PROPOSED SYSTEM

The proposed system is an Online Cruise Booking System that provides a complete digital solution for booking cruises, hotels, and activities. It offers a single platform where users can perform all booking-related tasks easily and securely. The system allows users to register, log in, search for cruises, view details, and make bookings online.

The proposed system provides real-time availability, pricing, and booking confirmation. It supports secure online payments through integrated payment gateways. After successful booking, users receive instant email notifications, which improves communication and trust.

The admin panel allows administrators to manage cruises, hotels, users, and payments efficiently. All data is stored securely in a database, reducing errors and improving data management. The proposed system saves time, reduces cost, increases accuracy, and enhances user experience compared to the existing system.

1.3 SCOPE OF SYSTEM

The scope of the Online Cruise Booking System is wide and covers all major functionalities required for cruise travel booking. The system allows users to search for cruises based on destination, date, and budget. Users can also view and book nearby hotels and onboard activities along with the cruise.

The system supports secure user authentication and payment processing. It provides booking history and confirmation details to users. The admin can add, update, or remove cruise packages, hotel details, and offers. The system is designed to be scalable so that more features can be added in the future.

The scope is limited to web-based access and does not include a mobile application in the current version. However, the system can be extended in the future to support mobile platforms and additional services.

1.4 OBJECTIVES

The main objective of the Online Cruise Booking System is to provide a simple, secure, and efficient platform for booking cruise trips online. The system aims to reduce manual work and eliminate dependency on travel agents.

Other objectives include improving user experience, providing real-time booking information, ensuring secure payment processing, and maintaining accurate booking records. The system also aims to help administrators manage bookings and users easily.

Overall, the objective is to develop a reliable full stack web application that meets user needs and follows industry standards.

CHAPTER – 2

PROPOSED SYSTEM REQUIREMENT GATHERING

2.1 STAKEHOLDER OF SYSTEM

Stakeholders are individuals or groups who are directly or indirectly involved with the system. The main stakeholders of the Online Cruise Booking System include users, administrators, cruise operators, hotel partners, and payment service providers.

Users are the primary stakeholders who use the system to search and book cruises. Administrators manage the system data and operations. Cruise operators and hotel partners provide service data. Payment gateways ensure secure transactions.

Each stakeholder has different expectations, and the system is designed to fulfill their requirements effectively.

2.2 REQUIREMENT GATHERING TECHNIQUE USED

Requirement gathering is an important step in system development. Various techniques were used to collect requirements for the Online Cruise Booking System. These include interviews with mentors and users, studying existing booking platforms, and analyzing documentation.

Observation was also used to understand user behavior and common problems faced in traditional booking systems. These techniques helped in understanding functional and non-functional requirements clearly.

Proper requirement gathering ensured that the system meets user expectations and avoids unnecessary features.

2.3 CONSOLIDATED LIST OF REQUIREMENT

The consolidated list of requirements includes both functional and non-functional requirements. Functional requirements include user registration, login, cruise search, booking, payment, and email notification. Admin functionalities include managing users, cruises, and bookings.

Non-functional requirements include system security, performance, usability, and availability. The system must be fast, secure, and easy to use. It should handle multiple users simultaneously without performance issues.

This consolidated list acts as a foundation for system design and development.

2.4 PROJECT DEFINITION

The OceanTrip is a full stack web application developed to automate the process of cruise travel booking. The project uses frontend technologies for user interaction and backend technologies for business logic and data management.

The system provides an integrated platform for users to plan and book their cruise trips easily. It improves efficiency, accuracy, and user satisfaction. The project demonstrates the practical implementation of full stack development concepts and real-world problem-solving.

CHAPTER – 3

SYSTEM MANAGEMENT AND PLANNING

3.1 FEASIBILITY STUDY

A feasibility study is conducted to determine whether the proposed system is practical and achievable. It helps in analyzing different aspects such as technical resources, cost, and operational usability before starting system development. For the Online Cruise Booking System, a feasibility study was carried out to ensure that the system can be developed successfully using available resources.

The feasibility study helps in identifying possible risks and challenges at an early stage. It also ensures that the system will meet user requirements within time and budget constraints. The feasibility study for this project is divided into three main types: technical feasibility, economic feasibility, and operational feasibility.

3.1.1 TECHNICAL FEASIBILITY

Technical feasibility checks whether the required technology and technical skills are available to develop the system. The Online Cruise Booking System is technically feasible because it is developed using widely used and reliable technologies such as Python, Django, HTML, CSS, JavaScript, and MySQL.

These technologies are open-source, well-documented, and supported by a large developer community. The development team has sufficient knowledge and experience in using these technologies. The system does not require any special or expensive hardware, making it easy to implement.

The backend framework Django provides built-in security, authentication, and database management features, which makes development faster and safer. MySQL efficiently handles large amounts of data such as user details and booking records. Therefore, from a technical point of view, the system is feasible and can be implemented successfully.

3.1.2 ECONOMIC FEASIBILITY

Economic feasibility evaluates whether the system is cost-effective. The Online Cruise Booking System is economically feasible because it uses open-source software tools, which significantly reduce development and licensing costs.

No expensive software or infrastructure is required to develop the system. The main costs involved are related to development time, internet usage, and minimal hosting expenses. Since the system is web-based, it reduces paperwork and manual labor, which helps in saving operational costs in the long run.

The system also has the potential to generate revenue through booking services and partnerships. Therefore, the benefits of the system outweigh the costs involved, making it economically feasible.

3.1.3 OPERATIONAL FEASIBILITY

Operational feasibility checks whether the system will be accepted and used effectively by users. The Online Cruise Booking System is designed to be user-friendly and easy to operate. The interface is simple, clear, and interactive, allowing users to complete bookings without technical knowledge.

The system provides step-by-step booking processes, clear instructions, and instant confirmations, which improve user satisfaction. Administrators can manage data easily through the admin panel without requiring complex training.

The system reduces manual work and increases efficiency, making it suitable for real-world use. Therefore, the system is operationally feasible and can be successfully adopted by users and administrators.

3.2 HARDWARE – SOFTWARE REQUIREMENT

The Online Cruise Booking System requires minimal hardware and software resources, making it easy to deploy and maintain.

Hardware Requirements:

- Laptop or Desktop Computer
- Minimum 8 GB RAM
- Intel i3/i5 or equivalent processor
- Stable Internet Connection

Software Requirements:

- Operating System: Windows / Linux
- Programming Language: Python 3.10 or higher
- Backend Framework: Django
- Frontend Technologies: HTML, CSS, JavaScript, Bootstrap
- Database: MySQL
- Development Tools: Visual Studio Code, Web Browser

These requirements are easily available and affordable, making the system suitable for implementation.

3.3 SYSTEM PLANNING

System planning is an important phase that defines how the project will be executed. Proper planning helps in completing the project on time and within scope. For the Online Cruise Booking System, planning was done by dividing the project into small tasks and assigning timelines.

System planning includes defining project activities, scheduling tasks, allocating resources, and monitoring progress. Tools like Work Breakdown

Structure (WBS) and Gantt Chart were used to organize and manage project development effectively.

Good system planning ensures smooth development, reduces delays, and improves project quality.

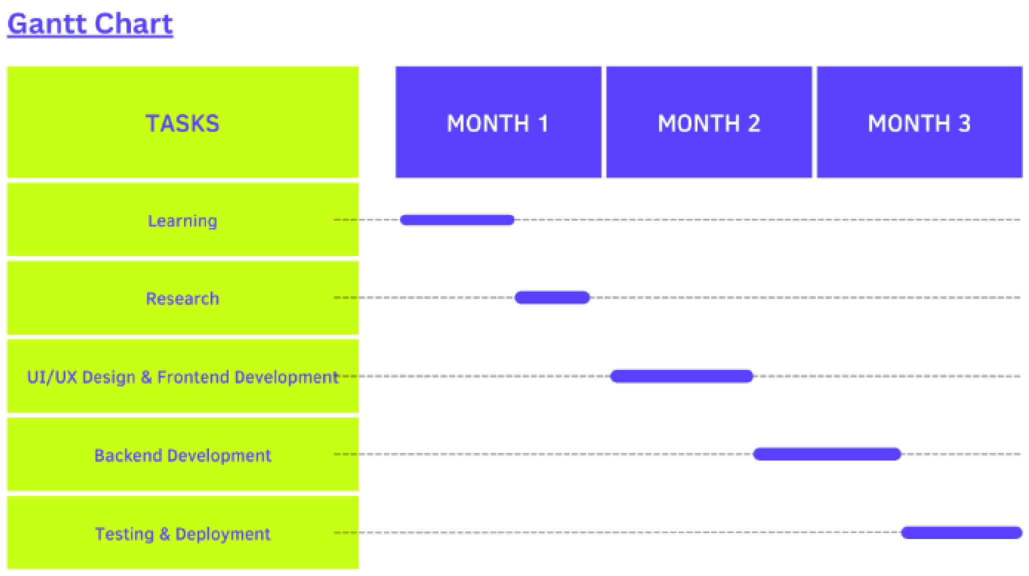
3.3.1 WORK BREAKDOWN STRUCTURE

Work Breakdown Structure (WBS) is a method of breaking a project into smaller, manageable tasks. For the Online Cruise Booking System, the project was divided into different phases such as requirement analysis, design, development, testing, and deployment.

Each phase was further divided into sub-tasks like frontend development, backend coding, database design, and testing. This helped in understanding the workload clearly and managing tasks efficiently.

WBS makes it easier to track progress and identify problems at an early stage. It also helps in assigning responsibilities and maintaining control over the project.

3.3.2 GANTT CHART



3.4 PROCESS MODEL

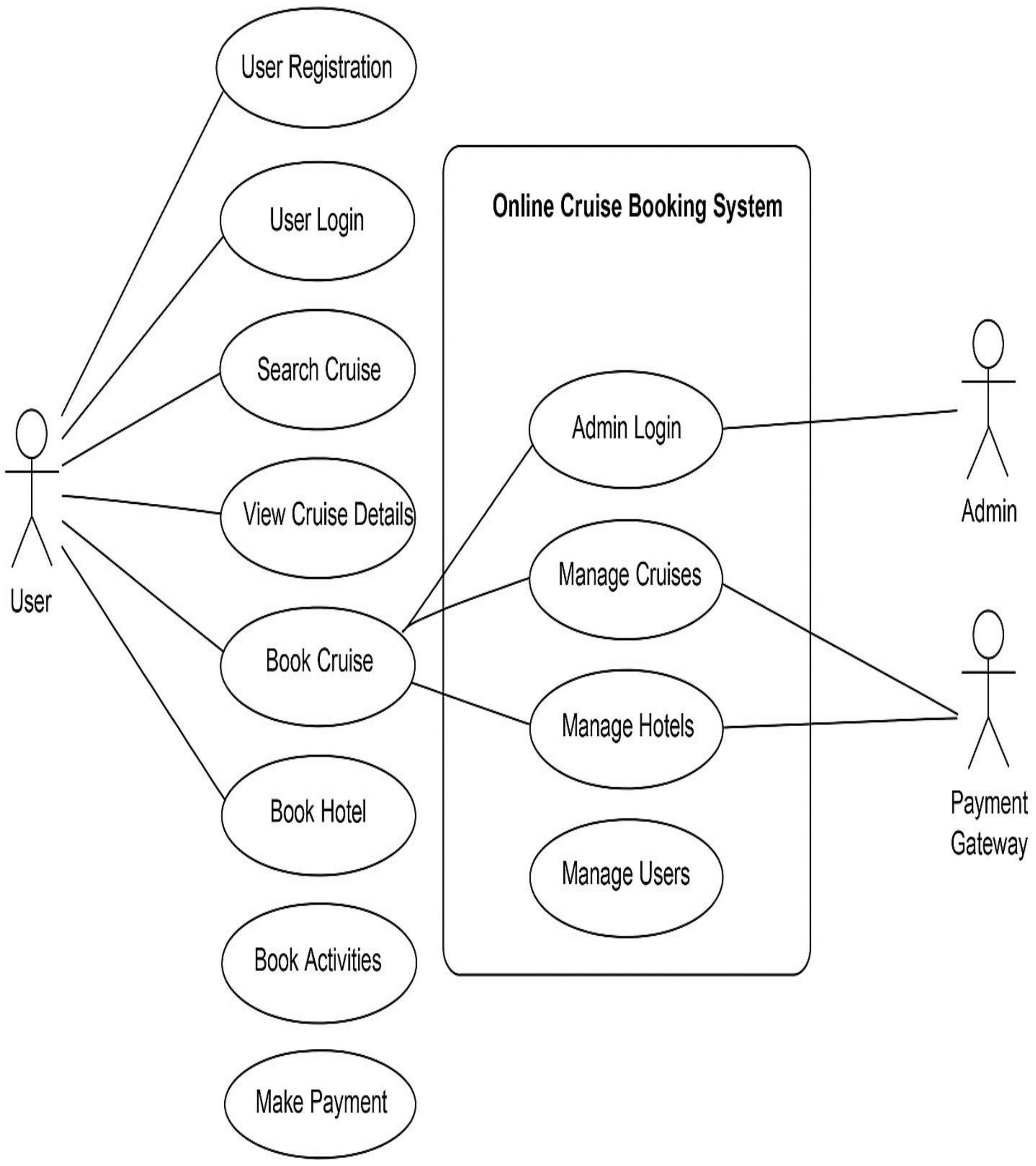
Work Breakdown Structure (WBS) is a method of breaking a project into smaller, manageable tasks. For the Online Cruise Booking System, the project was divided into different phases such as requirement analysis, design, development, testing, and deployment.

Each phase was further divided into sub-tasks like frontend development, backend coding, database design, and testing. This helped in understanding the workload clearly and managing tasks efficiently.

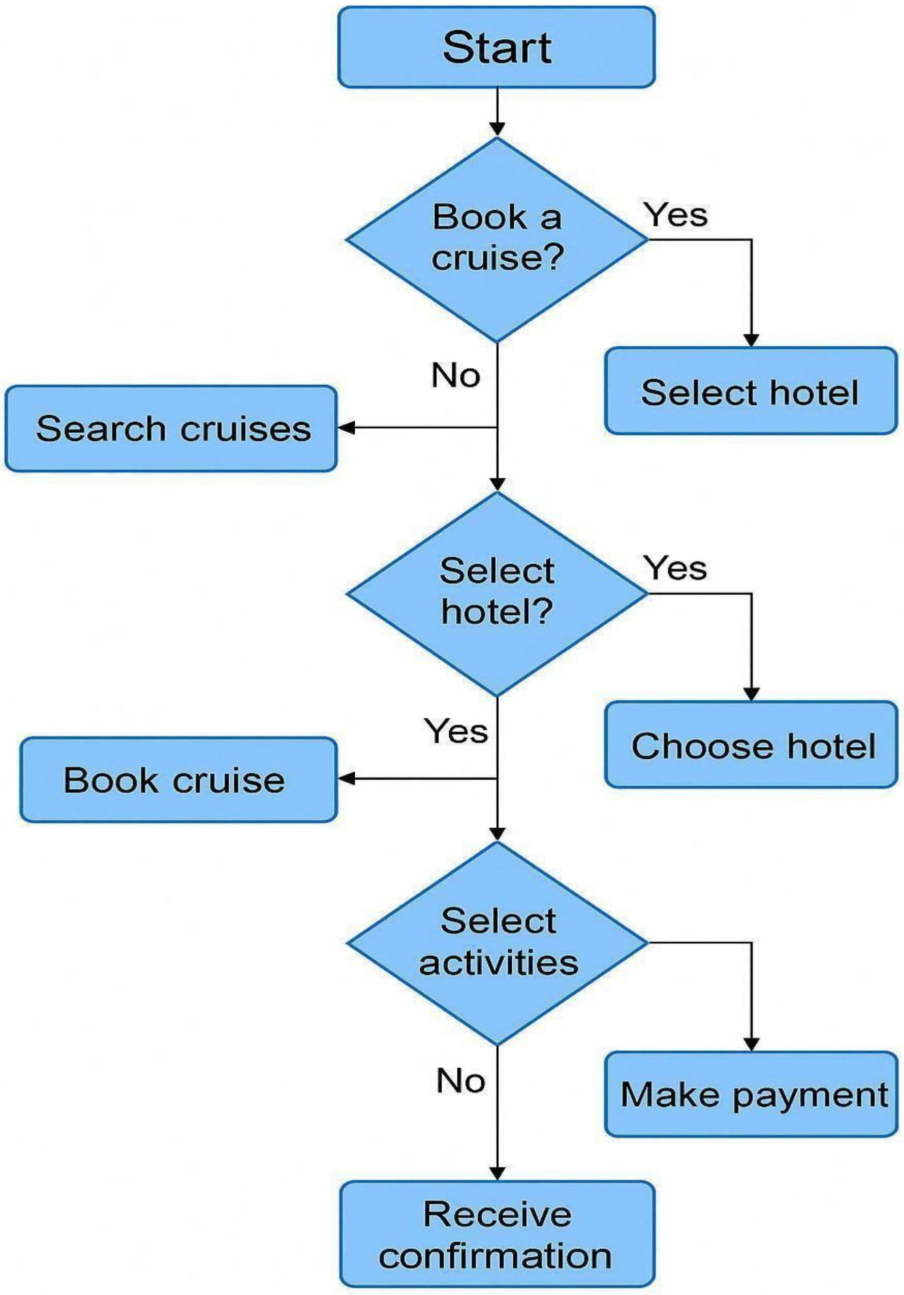
WBS makes it easier to track progress and identify problems at an early stage. It also helps in assigning responsibilities and maintaining control over the project.

CHAPTER - 4 SYSTEM ANALYSIS AND DESIGN

4.1 UML (UNIFIED MODELING LANGUAGE) / DFD



4.2 SYSTEM FLOW DIAGRAM



4.3 DATA DICTIONARY

1. USER TABLE

Field Name	Data Type	Size	Description
user_id	INT	10	Unique ID for each user
name	VARCHAR	50	Name of the user
email	VARCHAR	100	Email address of user
password	VARCHAR	255	Encrypted password
phone	VARCHAR	15	User contact number
created_at	DATETIME	—	Account creation date

2. ADMIN TABLE

Field Name	Data Type	Size	Description
admin_id	INT	10	Unique admin ID
username	VARCHAR	50	Admin username
password	VARCHAR	255	Encrypted admin password
role	VARCHAR	20	Admin role

3. CRUISE TABLE

Field Name	Data Type	Size	Description
cruise_id	INT	10	Unique cruise ID
cruise_name	VARCHAR	100	Name of cruise
destination	VARCHAR	100	Cruise destination
price	DECIMAL	—	Cruise price
duration	INT	—	Cruise duration (days)
availability	INT	—	Available seats

4. HOTEL TABLE

Field Name	Data Type	Size	Description
hotel_id	INT	10	Unique hotel ID
hotel_name	VARCHAR	100	Name of hotel
rating	INT	—	Hotel rating (3–5 star)
price_per_night	DECIMAL	—	Price per night
location	VARCHAR	100	Hotel location

5. ACTIVITY TABLE

Field Name	Data Type	Size	Description
activity_id	INT	10	Unique activity ID
activity_name	VARCHAR	100	Name of activity
price	DECIMAL	—	Activity cost
availability	VARCHAR	20	Available or not

6. BOOKING TABLE

Field Name	**Data Type**	**Size**	**Description**
booking_id	INT	10	Unique booking ID
user_id	INT	10	ID of user
booking_date	DATE	—	Date of booking
total_amount	DECIMAL	—	Total booking amount
status	VARCHAR	20	Booking status

7. PAYMENT TABLE

Field Name	**Data Type**	**Size**	**Description**
payment_id	INT	10	Unique payment ID
booking_id	INT	10	Related booking ID
payment_method	VARCHAR	50	Payment type
payment_status	VARCHAR	20	Success / Failed
payment_date	DATETIME	—	Payment date

4.4 USER INTERFACE

- Search Bar Interaction: Users enter their destination, preferred travel dates, and cruise type. The advanced search component filters through a curated list of 3-star to 5-star hotels, luxury cruise lines, and local activities based on their preferences.
- Resume Upload (Optional but Powerful): For users exploring job-based cruise packages or relocations, the resume upload section uses the intelligent parser (described in section 6.1) to personalize suggestions.
- Result Display: The frontend renders a list of filtered options using React components with real-time sorting, filtering, and availability status via asynchronous API calls.
- Detailed View Pages: Users can click into a cruise, hotel, or activity to view: HD images and videos , Pricing ,Reviews ,Location on map (via embedded Map API)
- Booking Cart: A multi-selection cart lets users add cruises, hotels, and activities together and proceed to checkout as a combined package.
- Secure Payment Workflow: Integrated payment gateways ensure encrypted transactions. After confirmation, users receive: Email/SMS alerts.

4.5 SYSTEM NAVIGATION

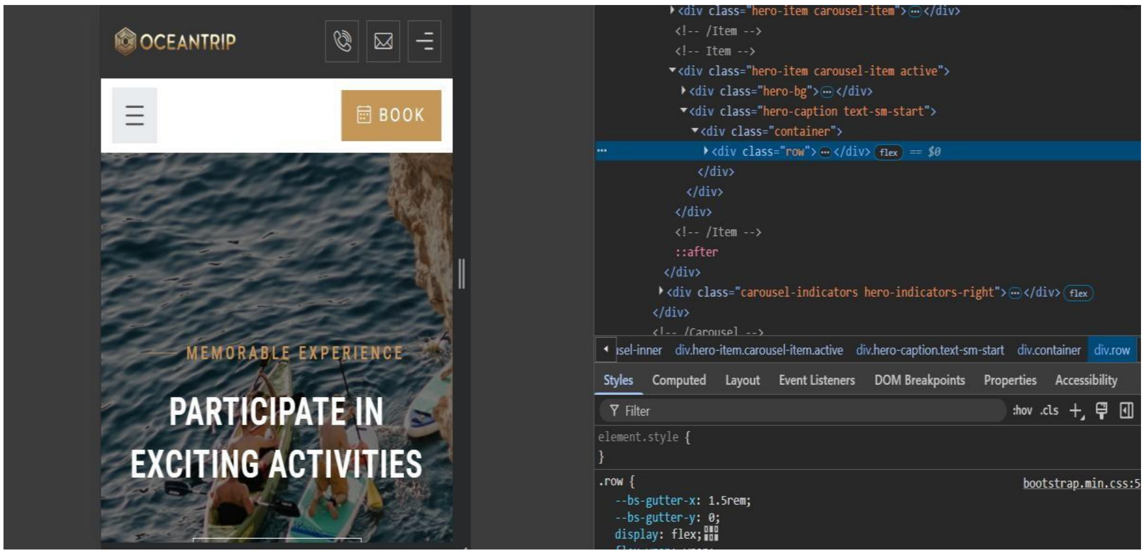
Oceantrip is built as a modular system where each component communicates reliably across clearly defined APIs.

- Backend: Django
  - Handles resume parsing logic, authentication, database operations, and job matching.
  - APIs return structured JSON data to the frontend via RESTful endpoints.
- Frontend: Html , Css , Js
  - Responsive and mobile-friendly UI with fast interactions.
- Database: MYSQL
  - Stores user credentials, parsed resumes, matched jobs, and interaction history.

CHAPTER -5 INPUT / OUTPUT DESIGN

UI Enhancements & Responsive Design

- Improved overall UI using custom CSS and Bootstrap utilities for spacing, typography, and layout.


OCEANTRIP
BOOK
MEMORABLE EXPERIENCE
PARTICIPATE IN EXCITING ACTIVITIES

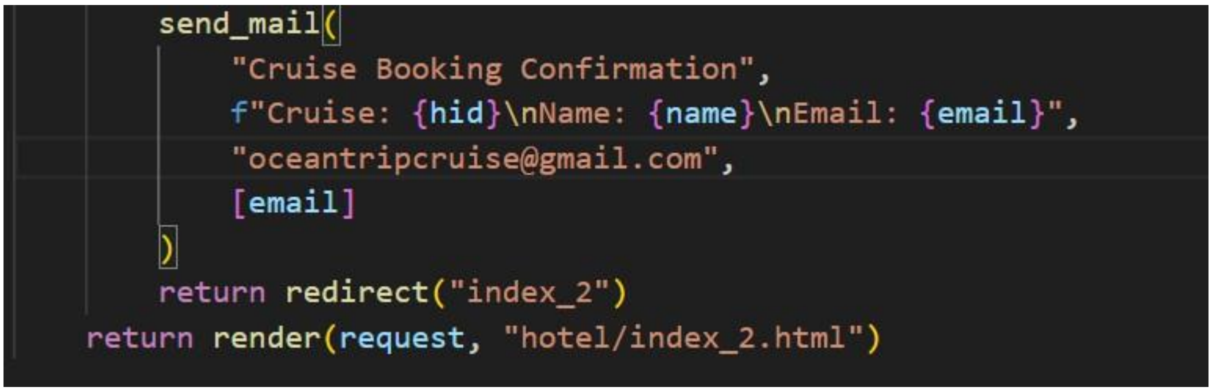
```
▸<div class="hero-item carousel-item">⋯</div>
 <!-- /Item -->
 <!-- Item -->
▾<div class="hero-item carousel-item active">
  ▸<div class="hero-bg">⋯</div>
  ▾<div class="hero-caption text-sm-start">
    ▾<div class="container">
      ▸<div class="row">⋯</div> flex == $0
     </div>
   </div>
 </div>
 <!-- /Item -->
 ::after
</div>
▸<div class="carousel-indicators hero-indicators-right">⋯</div> flex
</div>
<!-- /Carousel -->
```

isel-inner div.hero-item.carousel-item.active div.hero-caption.text-sm-start div.container div.row
Styles Computed Layout Event Listeners DOM Breakpoints Properties Accessibility
Filter :hov .cls

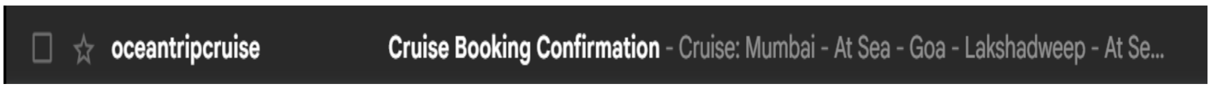
```
element.style {
}
.row {                                    bootstrap.min.css:5
  --bs-gutter-x: 1.5rem;
  --bs-gutter-y: 0;
  display: flex;
```

Email Notification

- Configured SMTP using Gmail settings in Django. Created email templates.
- Sent booking confirmation emails with booking details.



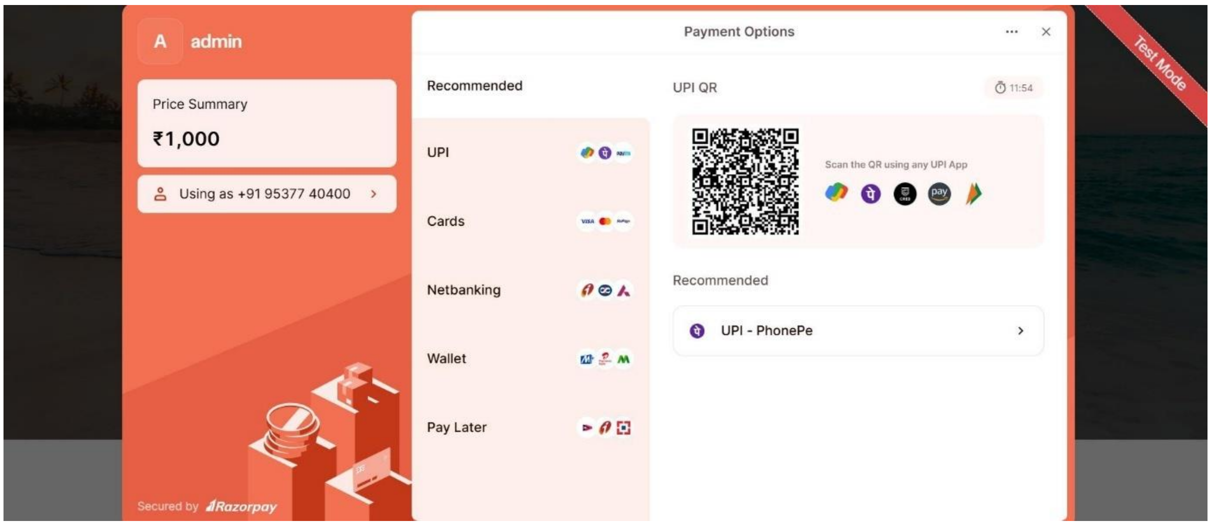
```
        send_mail(
            "Cruise Booking Confirmation",
            f"Cruise: {hid}\nName: {name}\nEmail: {email}",
            "oceantripcruise@gmail.com",
            [email]
        )
        return redirect("index_2")
    return render(request, "hotel/index_2.html")
```

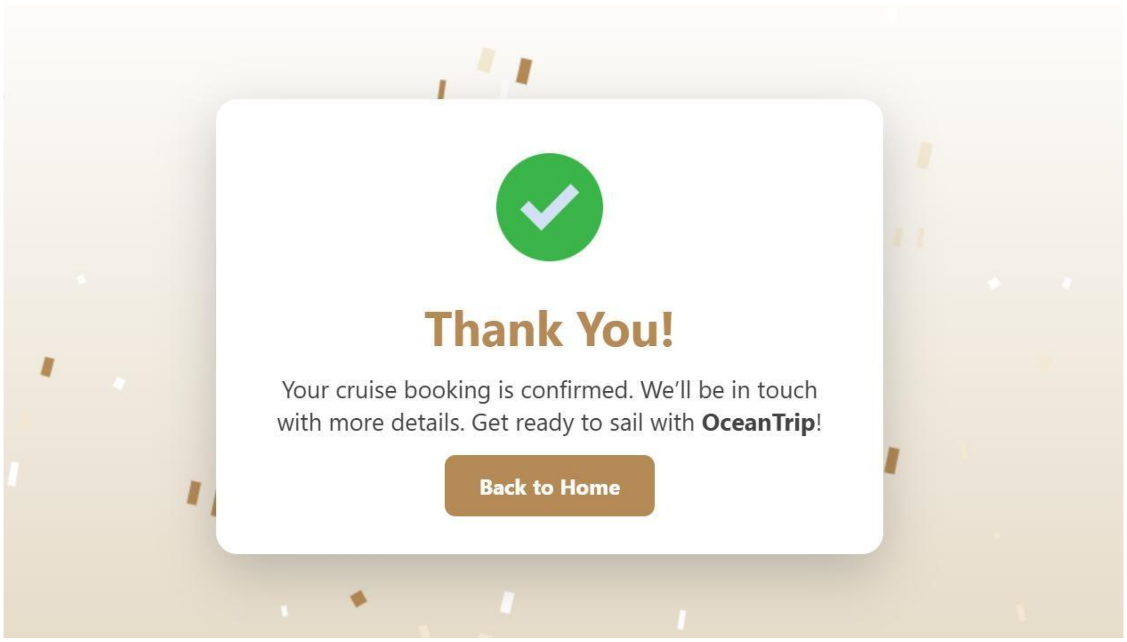


oceantripcruise   **Cruise Booking Confirmation** - Cruise: Mumbai - At Sea - Goa - Lakshadweep - At Se...

Payment Gateway Integration

- Integrated a dummy payment gateway (Razorpay). Created payment form and success page.



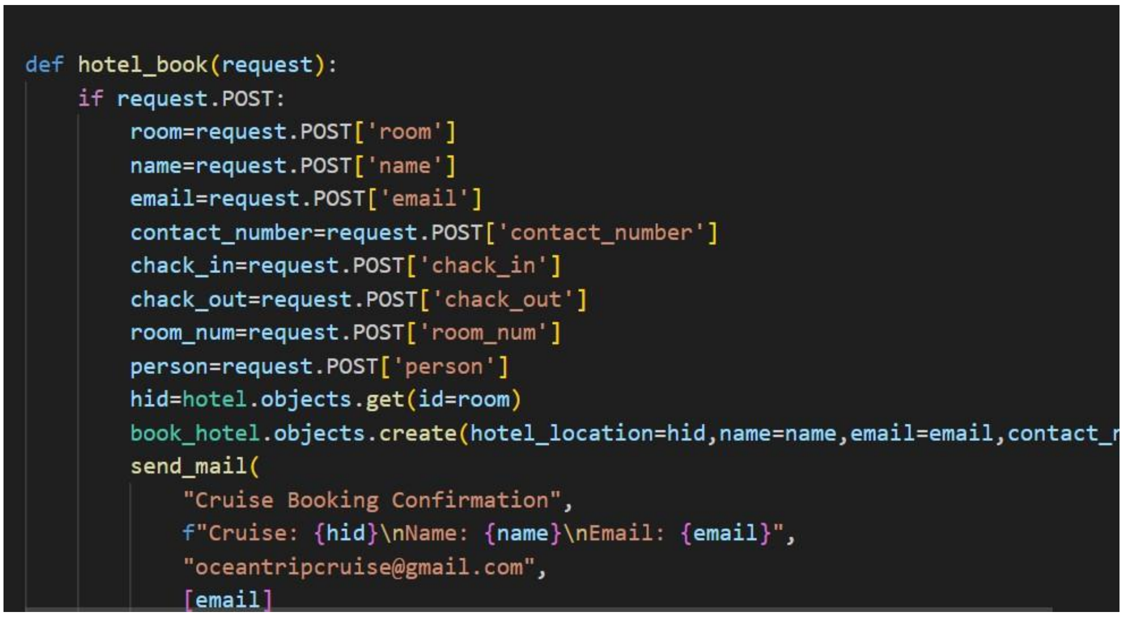


Thank You!

Your cruise booking is confirmed. We'll be in touch with more details. Get ready to sail with **OceanTrip**!

Booking System

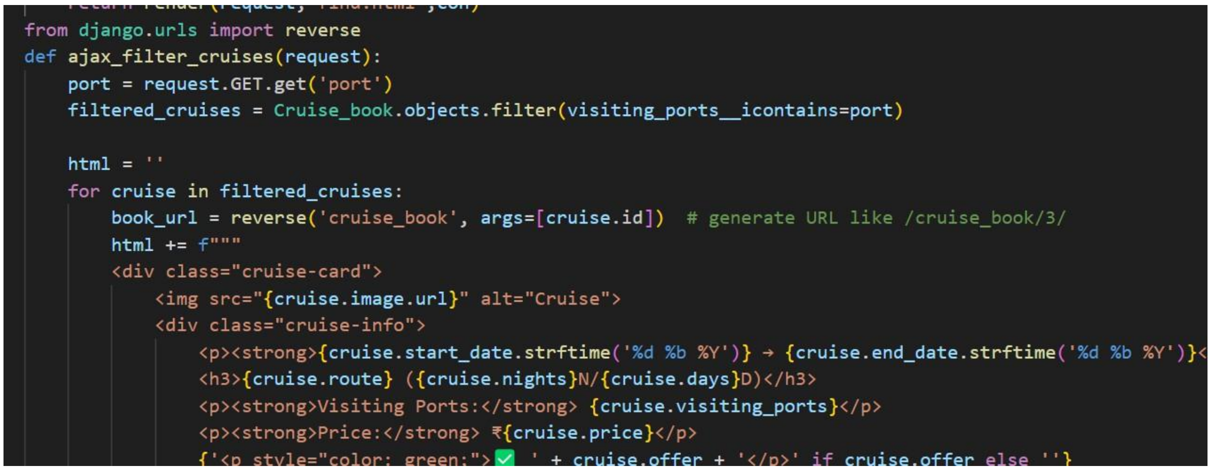
- Created a cruise booking form using Django forms. Users could choose dates, number of people, and cruise type.
- Extended booking form to include hotel and activities. Ensured data is captured correctly in the Booking model.



```python
def hotel_book(request):
    if request.POST:
        room=request.POST['room']
        name=request.POST['name']
        email=request.POST['email']
        contact_number=request.POST['contact_number']
        chack_in=request.POST['chack_in']
        chack_out=request.POST['chack_out']
        room_num=request.POST['room_num']
        person=request.POST['person']
        hid=hotel.objects.get(id=room)
        book_hotel.objects.create(hotel_location=hid,name=name,email=email,contact_r
        send_mail(
            "Cruise Booking Confirmation",
            f"Cruise: {hid}\nName: {name}\nEmail: {email}",
            "oceantripcruise@gmail.com",
            [email]
```

Search & Filters

- Created a search bar to search cruises by name, location, or port using icontains filter.

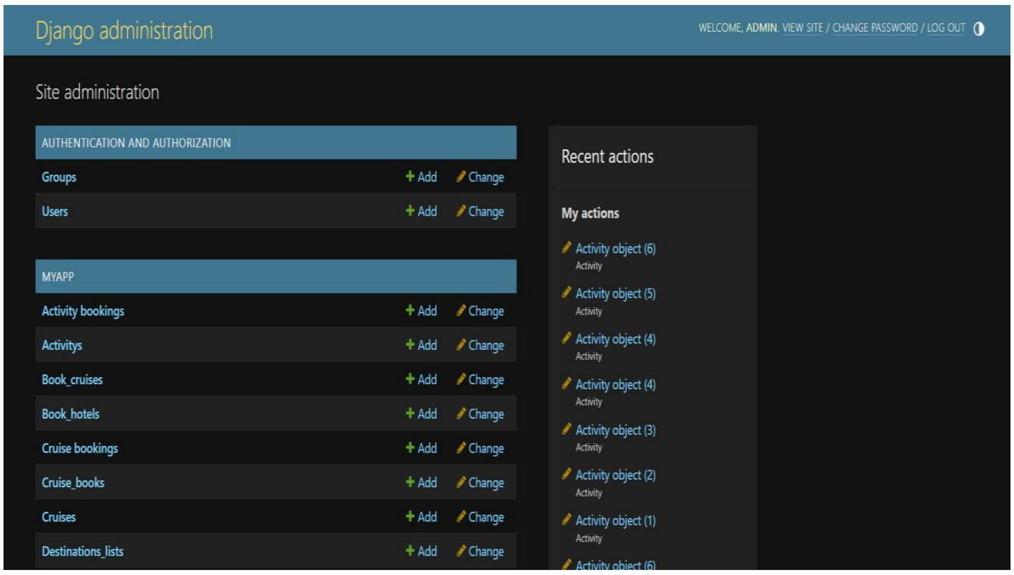


```python
    return render(request,'find.html',con)
from django.urls import reverse
def ajax_filter_cruises(request):
    port = request.GET.get('port')
    filtered_cruises = Cruise_book.objects.filter(visiting_ports__icontains=port)

    html = ''
    for cruise in filtered_cruises:
        book_url = reverse('cruise_book', args=[cruise.id])  # generate URL like /cruise_book/3/
        html += f"""
        <div class="cruise-card">
            <img src="{cruise.image.url}" alt="Cruise">
            <div class="cruise-info">
                <p><strong>{cruise.start_date.strftime('%d %b %Y')} → {cruise.end_date.strftime('%d %b %Y')}<
                <h3>{cruise.route} ({cruise.nights}N/{cruise.days}D)</h3>
                <p><strong>Visiting Ports:</strong> {cruise.visiting_ports}</p>
                <p><strong>Price:</strong> ₹{cruise.price}</p>
                {'<p style="color: green;">✅ ' + cruise.offer + '</p>' if cruise.offer else ''}
```

Database Models & Admin

- Designed core models: Cruise, Hotel, Booking, Port, and Activity. Defined fields like name, description, price, ratings, etc



Django administration

WELCOME, ADMIN. VIEW SITE / CHANGE PASSWORD / LOG OUT

Site administration

AUTHENTICATION AND AUTHORIZATION		
Groups	+ Add	Change
Users	+ Add	Change

MYAPP		
Activity bookings	+ Add	Change
Activitys	+ Add	Change
Book_cruises	+ Add	Change
Book_hotels	+ Add	Change
Cruise bookings	+ Add	Change
Cruise_books	+ Add	Change
Cruises	+ Add	Change
Destinations_lists	+ Add	Change

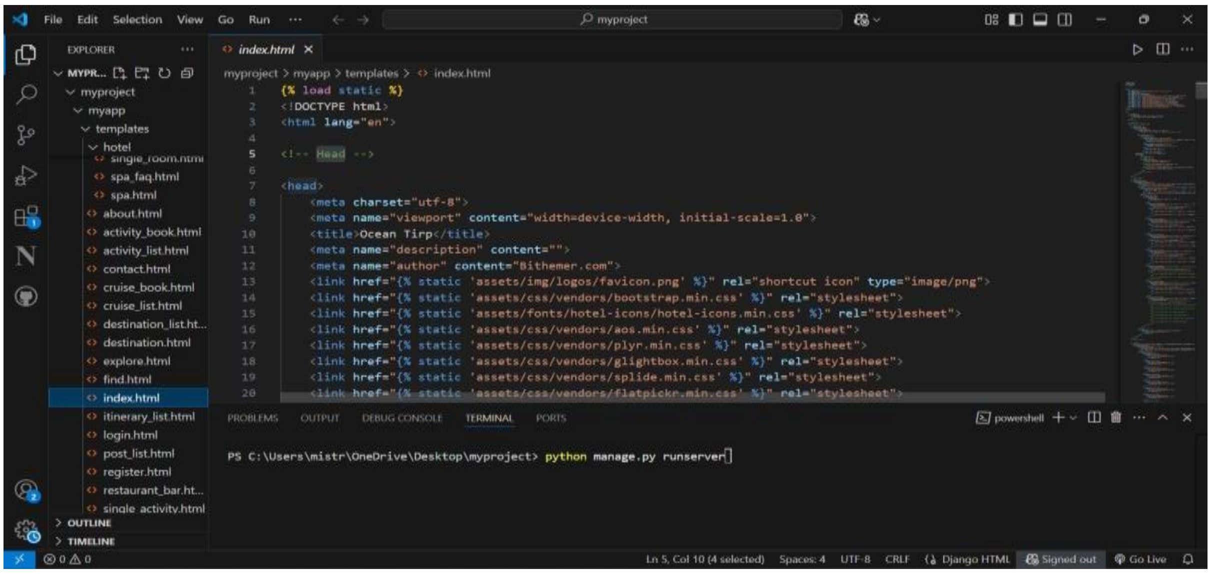
Recent actions

My actions

- Activity object (6) Activity
- Activity object (5) Activity
- Activity object (4) Activity
- Activity object (4) Activity
- Activity object (3) Activity
- Activity object (2) Activity
- Activity object (1) Activity
- Activity object (6)

Templates & Navigation

- Created multiple pages like About, Services, and Contact using views and mapped them with URLs. Learned URL dispatcher in Django.
- Implemented base layout using template inheritance ({% extends 'base.html' %}) and used {% block content %} to inject dynamic data.



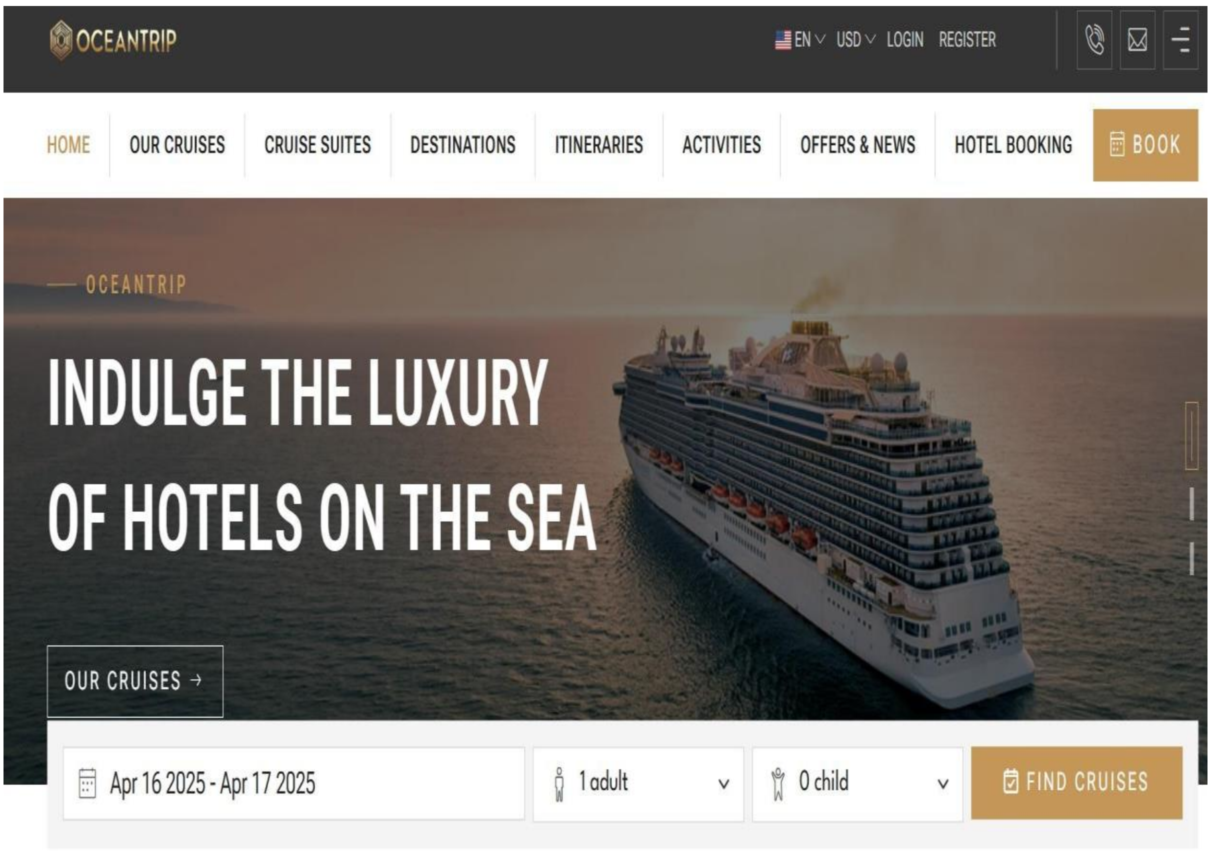
```
{% load static %}
<!DOCTYPE html>
<html lang="en">

<!-- Head -->

<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Ocean Tirp</title>
    <meta name="description" content="">
    <meta name="author" content="Bithemer.com">
    <link href="{% static 'assets/img/logos/favicon.png' %}" rel="shortcut icon" type="image/png">
    <link href="{% static 'assets/css/vendors/bootstrap.min.css' %}" rel="stylesheet">
    <link href="{% static 'assets/fonts/hotel-icons/hotel-icons.min.css' %}" rel="stylesheet">
    <link href="{% static 'assets/css/vendors/aos.min.css' %}" rel="stylesheet">
    <link href="{% static 'assets/css/vendors/plyr.min.css' %}" rel="stylesheet">
    <link href="{% static 'assets/css/vendors/glightbox.min.css' %}" rel="stylesheet">
    <link href="{% static 'assets/css/vendors/splide.min.css' %}" rel="stylesheet">
    <link href="{% static 'assets/css/vendors/flatpickr.min.css' %}" rel="stylesheet">
```

```
PS C:\Users\mistr\OneDrive\Desktop\myproject> python manage.py runserver
```

- Created separate folders for static files and media uploads. Configured URLs for static and media files for development.
- Refined UI design of pages. Added hover effects, carousel, and responsive grid layout to improve user experience.



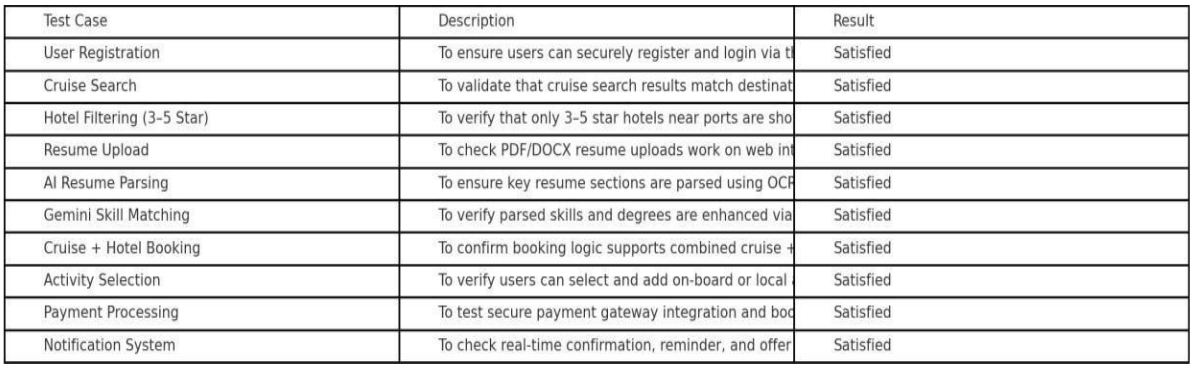
CHAPTER -6 TESTING

1.1 *TESTING STRATEGIES*

The OceanTrip system was tested using unit, integration, system, and user acceptance testing to ensure smooth functionality. Performance and security tests were also conducted to verify speed and data protection. These testing strategies ensured the application worked correctly, was user-friendly, and could handle multiple users securely and efficiently.

1.1.1 Functional Testing

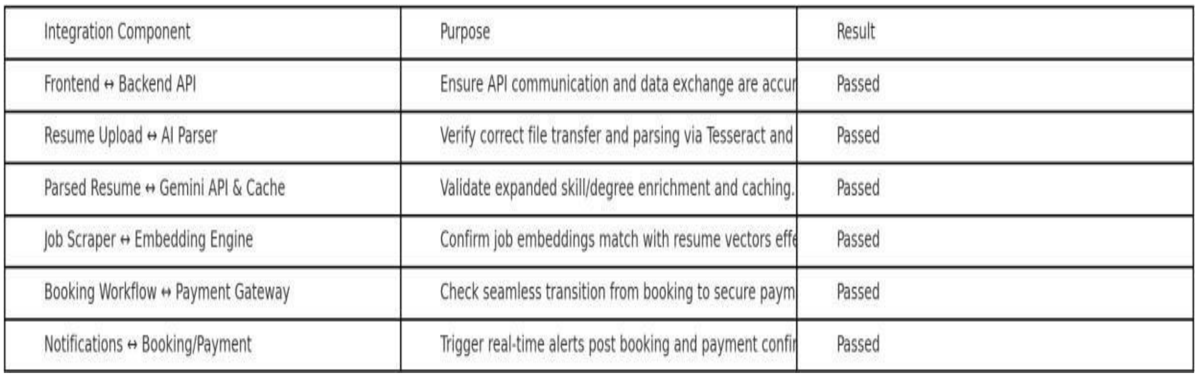
Goal: Verify the core functionalities such as resume upload, parsing, and job recommendation delivery.



Test Case	Description	Result
User Registration	To ensure users can securely register and login via t	Satisfied
Cruise Search	To validate that cruise search results match destinat	Satisfied
Hotel Filtering (3-5 Star)	To verify that only 3-5 star hotels near ports are sho	Satisfied
Resume Upload	To check PDF/DOCX resume uploads work on web int	Satisfied
AI Resume Parsing	To ensure key resume sections are parsed using OC	Satisfied
Gemini Skill Matching	To verify parsed skills and degrees are enhanced via	Satisfied
Cruise + Hotel Booking	To confirm booking logic supports combined cruise +	Satisfied
Activity Selection	To verify users can select and add on-board or local	Satisfied
Payment Processing	To test secure payment gateway integration and bo	Satisfied
Notification System	To check real-time confirmation, reminder, and offer	Satisfied

Figure 7.1 Function Testing Results

1.1.1 Integration Testing

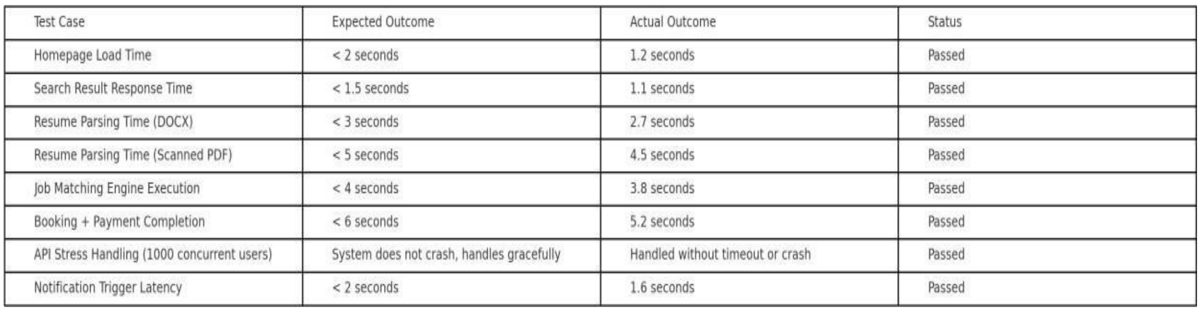


Integration Component	Purpose	Result
Frontend ↔ Backend API	Ensure API communication and data exchange are accu	Passed
Resume Upload ↔ AI Parser	Verify correct file transfer and parsing via Tesseract and	Passed
Parsed Resume ↔ Gemini API & Cache	Validate expanded skill/degree enrichment and caching.	Passed
Job Scraper ↔ Embedding Engine	Confirm job embeddings match with resume vectors eff	Passed
Booking Workflow ↔ Payment Gateway	Check seamless transition from booking to secure paym	Passed
Notifications ↔ Booking/Payment	Trigger real-time alerts post booking and payment confi	Passed

Figure 7.2 Integration Testing Results

.

1.1.2 Performance Testing



Test Case	Expected Outcome	Actual Outcome	Status
Homepage Load Time	< 2 seconds	1.2 seconds	Passed
Search Result Response Time	< 1.5 seconds	1.1 seconds	Passed
Resume Parsing Time (DOCX)	< 3 seconds	2.7 seconds	Passed
Resume Parsing Time (Scanned PDF)	< 5 seconds	4.5 seconds	Passed
Job Matching Engine Execution	< 4 seconds	3.8 seconds	Passed
Booking + Payment Completion	< 6 seconds	5.2 seconds	Passed
API Stress Handling (1000 concurrent users)	System does not crash, handles gracefully	Handled without timeout or crash	Passed
Notification Trigger Latency	< 2 seconds	1.6 seconds	Passed

Figure 7.3 Performance Testing Results

CHAPTER – 7
SUMMARY

7.1 ASSUMPTIONS

Assumptions are the conditions that are considered to be true during the development of the system. The Online Cruise Booking System has been developed based on certain assumptions to ensure smooth functioning of the application.

It is assumed that users have basic knowledge of using the internet and web applications. Users are expected to have access to a stable internet connection to search and book cruises without interruption. It is also assumed that users will provide correct and valid information during registration and booking.

The system assumes that payment gateways are available and functioning properly at the time of transaction. It is also assumed that the server and database remain active and secure to store user and booking data. These assumptions help in defining the boundaries within which the system operates efficiently.

7.2 LIMITATIONS

Every system has certain limitations, and the Online Cruise Booking System is no exception. One of the main limitations is that the system is currently designed as a web-based application only and does not include a dedicated mobile application.

The system supports limited payment options depending on gateway availability. Real-time updates may sometimes face delays due to server or network issues. The system also depends on accurate data provided by cruise operators and hotel partners.

Another limitation is that the system does not support multi-language functionality in its current version. Advanced personalization and dynamic pricing features are also not included. These limitations can be addressed in future versions of the system.

7.3 FUTURE SCOPE

The Online Cruise Booking System has wide future scope and can be enhanced with additional features. A mobile application for Android and iOS platforms can be developed to increase accessibility. AI-based recommendation systems can be implemented to suggest cruises and hotels based on user preferences.

Multi-language support can be added to make the system usable for international users. Chatbot integration can be introduced to provide real-time customer support. Integration with GPS and maps can help users view port locations easily.

Advanced analytics, dynamic pricing, loyalty programs, and promotional offers can also be included. These enhancements will improve user experience and system efficiency in the future.

7.4 CONCLUSION

The Online Cruise Booking System is a complete full stack web application developed to simplify the cruise booking process. The system provides a centralized platform for booking cruises, hotels, and onboard activities securely and efficiently.

By automating the booking process, the system reduces manual effort, saves time, and minimizes errors. The use of modern technologies such as Python Django, HTML, CSS, JavaScript, and MySQL ensures system reliability, scalability, and security.

Overall, the project successfully meets its objectives and demonstrates practical implementation of full stack development concepts. The system is user-friendly, efficient, and suitable for real-world use.

BIBLIOGRAPHY

1. Python Software Foundation, *Python Documentation*.
Available at: https://www.python.org
2. Django Software Foundation, *Django Framework Documentation*.
Available at: https://docs.djangoproject.com
3. MySQL Documentation, *MySQL Database Reference*.
Available at: https://dev.mysql.com/doc
4. Mozilla Developer Network (MDN), *HTML, CSS and JavaScript Documentation*.
Available at: https://developer.mozilla.org
5. Bootstrap Documentation, *Responsive Web Design Framework*.
Available at: https://getbootstrap.com

Thanks For Watching