UNIVERSITY OF CALOOCAN CITY
COLLEGE OF ENGINEERING
Phase 8A Package 11, Block 199, German Village, barangay 176 Bagong Silang, Caloocan City, 1428

Development of an Online Web-Based Booking System for Reservation and Resort Management

Gabijan, Rhovic M. - Leader
Balana, Jerkielle Roen O.
Balaoro, Judge Wayne B.
Barbas, Steven Jade P.
Dispo, Lei Andrew T.
Laput Mark Danielle E.

University of Caloocan City
Caloocan City

Approval Sheet

This design project entitled **“Development of an Online Web-Based Booking System for Reservation and Resort Management”** prepared by **Jerkielle Roen O. Balana, Judge Wayne B. Balaoro, Steven Jade P. Barbas, Rhovic M. Gabijan, Mark Danielle E. Laput** of the Computer Engineering Department, was examined and evaluated by the members of the Student Design Evaluation Panel and is hereby recommended for approval.

Engr. Maria Rizette H. Sayo
Facilitator

Student Design Evaluation Panel:

ENGR/DR. FIRSTNAME LASTNAME
Panel Member

ENGR/DR. FIRSTNAME LASTNAME
Panel Member

ENGR/DR. FIRSTNAME LASTNAME
Lead Panel

ENGR. MARIA CONCEPCION MIRABUENO
Program Chair

UNIVERSITY OF CALOOCAN CITY
Caloocan City

SOFTWARE DESIGN PROJECT INFORMATION

2nd Semester, SY 2025-2026

Student/Team **Team BLDG.**	Jerkielle Roen O. Balana Judge Wayne B. Balaoro Steven Jade P. Barbas Lei Andrew T. Dispo Rhovic M. Gabijan Mark Danielle E. Laput
Project Title	Development of an Online Web-Based Booking System for Reservation and Resort Management.
Project Concentration Area	Programming Logic and Design, Object-Oriented Programming, Data Structure and Algorithm, Database Management System
Design Objectives	The general objective of this project is to create a centralized, user-friendly, and responsive web-based booking system for **Palacio Feliz - Event and Private Resort**. This platform aims to highlight the resort's amenities and services while automating the reservation process to reduce manual workload and scheduling errors. **Specifically, it aims to:** ● **Design** a responsive user interface accessible on both desktop and mobile devices to showcase rooms, amenities, and real-time availability. ● **Develop** a secure customer interface that allows for online booking form submission and automated email confirmations. ● **Develop** an administrator dashboard for monitoring bookings, managing schedules, and organizing client data (add stats). ● **Develop** a notification system to provide automated reminders and status updates for both the client and the administrator. ● **Test and evaluate** the system's accuracy in handling real-time scheduling to ensure there are no overlapping reservations or data entry errors.
Constraints	
Constraint (Metric): Budget	The project is for academic purposes, so the development cost must be kept low. According to the Cost Analysis, the budget is limited to essential hosting and domain fees (approx. ₱200–₱300/month or

	₱3,000 for a .ph domain), with no budget for professional labor or premium software licenses.
Constraint (Metric): Timeline	The system must be fully deployed and functional by the end of the 2nd Semester (April 2026). This includes all phases: planning, design, development, and testing.
Constraint (Metric): Performance	The user flow is constrained to a "minimal click" design. A user must be able to complete a reservation in four primary steps or fewer to ensure high usability.
Constraint (Metric): Data Accuracy (Zero Double-Bookings)	The system must have a 100% accuracy rate in date validation. It is constrained by the requirement to instantly "lock" a date upon admin approval to prevent overlapping reservations.
Constraint (Metric): Accessibility	The system must be responsive and functional across at least two platform types: Desktop/Laptop (Windows/macOS) and Mobile (Android/iOS).
Other constraints: These constraints do not affect each design; therefore, these were not included in selecting the best design.	
Constraint: Development Tools	The team is restricted to using open-source technologies (Python/Django, HTML/CSS/JS, and VS Code) to maintain a zero-cost development environment.
Constraint: Legal/Contractual	All development must adhere to the signed contract, which limits the developers to academic use and grants the client full rights to the operational code upon completion.
Constraint: Network Dependency	The system requires an active internet connection to process real-time bookings and send SMS/Email notifications; offline functionality is outside the current project scope.
Constraint: Data Privacy	The system must comply with basic data handling ethics to protect guest personal information and payment proofs uploaded to the server.
Standards	
ISO/IEC 25010:2011 (Systems and Software Quality Models)	**Definition:** This standard defines the quality characteristics of software, specifically focusing on Functional Suitability and Usability. **Application:** This standard was used to validate the system's ability to prevent double-bookings (Functional Suitability) and to ensure the dashboard is intuitive for non-technical resort staff (Usability).

W3C Web Standards (HTML5 & CSS3 Architecture)	**Definition:** Established by the World Wide Web Consortium, this standard defines the open technical specifications for the structural and presentational layers of web applications. **Application:** This standard was applied to the frontend development to guarantee cross-platform compatibility, ensuring the booking website renders correctly on both mobile phones and desktop computers.
ISO/IEC 27001:2022 (Information Security)	**Definition**: This standard provides the framework for an Information Security Management System (ISMS) to preserve the confidentiality and integrity of data. **Application**: This standard was used to implement Access Control protocols, ensuring that sensitive guest data (such as names and payment receipts) are accessible strictly by the authorized Administrator account.
PEP 8 (Python Enhancement Proposal 8)	**Definition**: This is the industry-standard style guide for Python code, providing conventions for indentation, naming, and structure. **Application**: This standard was applied to the Django backend code to ensure maintainability and readability for future developers or system upgrades.

For single pages, use "p." For multiple pages, use "pp."
Appendices should be italicized and referred to every time it is mentioned.

Abstract

This project addresses the need for enhanced comfort, security, and energy efficiency in residential environments by designing and implementing an intelligent home automation system leveraging Internet of Things (IoT) and machine learning techniques. The system integrates various smart sensors, actuators, and devices interconnected through a robust IoT framework. Machine learning algorithms enable the system to learn and adapt to residents' behaviors and preferences, providing personalized automation and control. Key features include automated lighting, climate control, security monitoring, and energy management. Performance evaluation through real-world scenarios demonstrates significant improvements in user convenience and energy savings. The project highlights the potential of combining IoT with machine learning to create an adaptive and efficient smart home environment, significantly enhancing the quality of life for residents by offering customized and efficient home automation solutions.

Keywords: *Home Automation, Internet of Things (IoT), Machine Learning, Smart Sensors, Energy Efficiency, Security Monitoring, Adaptive Control, Smart Home*

List of Tables

List of figures

List of abbreviation

Definition of terms

TABLE OF CONTENTS

CHAPTER 1: THE PROJECT AND ITS BACKGROUND

The hospitality and leisure industry in the Philippines is increasingly shifting toward digitalization to meet the demands of modern consumers who prioritize convenience and instant accessibility. **Palacio Feliz - Event and Private Resort**, located in Gaya-Gaya, San Jose del Monte, Bulacan, is a private destination offering amenities for events and personal stays. Currently, the resort manages its operations through traditional social media inquiries and manual recording.

This project, titled **"Development of an Online Web-Based Booking System for Reservation and Resort Management,"** is developed by the BLDG. Development Team as part of the requirements for the course CpE-201L (Software Design). The project aims to modernize the resort's operational workflow by providing a dedicated digital platform that handles reservations, showcases amenities, and automates administrative tasks, ensuring a professional and efficient experience for both the resort management and its client.

1.1 The Problem

The current manual reservation system of **Palacio Feliz - Event and Private Resort** relies on physical logbooks and social media inquiries, which creates a significant gap in operational efficiency. This traditional approach presents the following technical and operational challenges:

- **Scheduling Conflicts:** The lack of real-time validation leads to "pencil booking" errors, resulting in accidental double-bookings or overlapping schedules.
- **Data Vulnerability:** Critical reservation records and guest information are stored only in physical logbooks, making them susceptible to damage or permanent loss.
- **Response Latency:** Customers cannot check availability instantly, requiring a manual back-and-forth through social media that delays the booking process.
- **Inefficient Payment Tracking:** Proof of payment is handled through external apps and manually cross-referenced, increasing the risk of administrative oversight.
- **Limited Accessibility:** Without a centralized platform, the resort lacks a professional digital presence to automate inquiries and secure bookings outside of business hours.

1.2 The Client

The client for this project is **Palacio Feliz - Event and Private Resort**, a local leisure destination located at Evergreen Ave, Gaya-Gaya, San Jose del Monte, Bulacan. Managed by **Ms Christina Limin.**, the resort offers amenities such as private pools, event spaces, and overnight accommodations. Currently, the resort relies on manual inquiries through Facebook and physical logbook entries to manage guest stays. The client seeks a digital transition to professionalize their booking process and reach a wider customer.

Table 1-1. Client and Engineering Requirements / Considerations

Client Requirements / Considerations	Engineering Requirements / Considerations
The system can showcase the resort's	**Frontend Design:** Implementation of a

Client Requirements / Considerations	Engineering Requirements / Considerations
amenities, room photos, and current rates to potential guests.	responsive UI using HTML/CSS/JS with a dedicated Image Gallery and dynamic pricing display.
The system can allow guests to check if their desired dates are available without needing to message the staff.	**Real-time Database:** A backend logic using Python/Django to query the database and reflect live availability on a calendar interface.
The system can provide a way for guests to submit their personal details and proof of payment (receipts) online.	**Data Handling & Storage:** Integration of secure web forms with a File Upload field and a relational database (SQL) to store guest records.
The system can allow management to view all upcoming schedules and approve or cancel bookings easily.	**Admin Dashboard:** Development of a secure backend portal with CRUD (Create, Read, Update, Delete) capabilities for reservation management.
The system can notify both the owner and the guest once a reservation is officially confirmed.	**Automated Notifications:** Implementation of an SMTP server for Email alerts or an API for email notifications triggered by admin approval.

1.3 The Project

The project, titled **"Development of an Online Web-Based Booking System for Reservation and Resort Management,"** is a centralized reservation platform designed for **Palacio Feliz - Event and Private Resort** . The system automates the resort's operational workflow, facilitating a transition from a manual logbook recording system to a digital interface. Key features include a public-facing website that allows guests to view amenities, check real-time availability, and submit booking requests with proof-of-payment attachments. Simultaneously, it provides a secure administrator dashboard for the resort management to approve reservations, manage schedules, and send automated notifications, thereby eliminating double-bookings and improving data retrieval.

Web-Based Booking System for Palacio Feliz

This project focuses on the development of a centralized reservation and management platform titled **"Development of an Online Web-Based Booking System for Reservation and Resort Management"** The system is designed to automate the resort's operational workflow, transitioning from a manual logbook recording system to a digital interface. It features a public-facing website that allows guests to view amenities, check real-time availability, and submit booking requests with proof-of-payment attachments. Simultaneously, it provides a secure administrator dashboard for the resort management to approve reservations, manage schedules, and send automated notifications, thereby eliminating double-bookings and improving data retrieval.

Technologies Used To achieve these functionalities, the project utilizes the following software development tools and technologies:

- **Frontend Development:** The user interface is built using **HTML5, CSS3, and JavaScript** to ensure a responsive design that adapts to both desktop and mobile devices. This ensures potential guests can easily navigate the booking flow regardless of the device used.
- **Backend Framework:** The core logic of the system is developed using **Python** with the **Django Framework**. Django handles the URL routing, database management, and security features (such as Cross-Site Request Forgery protection) required for handling user data.
- **Database Management:** The system employs a relational database (integrated with Django) to store critical information, including guest profiles, reservation dates, and payment status records.
- **Deployment & Hosting:** For the academic prototype phase, the system is deployed using **PythonAnywhere**, utilizing its cloud-based server capabilities to host the web application and make it accessible online via a subdomain.
- **Development Environment:** The coding and debugging process is conducted using **Visual Studio Code (VS Code)**, serving as the primary Integrated Development Environment (IDE).

1.4 Project Objectives

The general objective of this project is to create a centralized, automated web-based booking system for **Palacio Feliz - Event and Private Resort**. This platform aims to modernize the resort's operational workflow by transitioning from manual logbook recording to a digital interface that showcases amenities, manages real-time availability, and streamlines the reservation process for both guests and management.

Specifically, the project aims to:

- **Design** a web-based software that has a responsive and user-friendly interface that displays the resort’s amenities, room, rates, and photo gallery across various devices.
- **Develop** a web-based booking module featuring:
    - Real-time availability checking and automated form submission.
    - A proof-of-payment upload feature for guest verification.
- **Develop** an administrator dashboard for resort management, including:
    - Features for monitoring bookings, managing schedules, and updating pricing.
    - A centralized customer information management system.

- **Develop** an automated notification system to provide instant booking confirmations and reminders via email.
- **Test and evaluate** the system's accuracy in managing schedules, preventing double-bookings, and maintaining data integrity between the user interface and the backend database.

1.5 Scope and Delimitations

The scope of this **Software Design Project** includes the design and implementation of a centralized **Web-Based Booking System** using **modern open-source web development technologies**. The system will integrate various modules to provide functionalities such as a public-facing website for showcasing amenities, a real-time availability calendar, a secure reservation form with proof-of-payment upload, and an administrator dashboard for schedule management. The target audience for this project includes the resort management of **Palacio Feliz - Event and Private Resort** and potential guests seeking a streamlined reservation experience. The project will be implemented as a responsive web application and will demonstrate the potential of digitizing manual logbook operations to improve operational efficiency. The system will be evaluated through functional testing to measure its performance in terms of scheduling accuracy, prevention of double-bookings, and user convenience.

The project is limited to the development of a **web-based application** accessible via browsers and does not include the creation of native mobile applications (APK or IPA) for Android or iOS. The system will only utilize **manual verification** for payments, requiring guests to upload screenshots of receipts, and does not include direct integration with payment gateway APIs (e.g., GCash, PayPal, or credit cards) for real-time transaction processing. The system requires an **active internet connection** to function and does not support offline data synchronization. While the system includes standard authentication for administrators, advanced cybersecurity measures such as penetration testing and protection against sophisticated distributed attacks are not covered.

1.6 Design Constraints

Accessibility:
Budget/Cost:
Legal Considerations:
Maintainability
Timeline/Schedule
Usability/Performance:
Data Accuracy
Development Tools (Not sure)
Network Dependency

Other Constraint:
- Sustainability

Functionality (Booking Error Rate) is the capability of the Web-Based Booking System to process reservation requests accurately without resulting in scheduling conflicts or data loss. The Booking Error Rate defines the frequency of failed transactions or instances where the system allows two guests to book the same slot. It is computed by subtracting the percentage of successful, non-conflicting bookings from the total attempts (1 - Accuracy). A higher error rate indicates lower functional performance and creates operational liabilities for the resort management. Therefore, the design with the lowest error rate is the winning design.

Other Constraints: These constraints do not affect each design choice equally; therefore, these were not included in the primary selection metric but are inherent requirements of the project.

- **Accessibility:** The system ensures accessibility by being available on any device with a web browser (mobile, tablet, or desktop), allowing users to access the platform regardless of their hardware.
- **Cost:** The project is bound by a "zero-cost" development budget, requiring the use of open-source technologies (Python/Django) and free-tier hosting services.
- **Legal considerations:** The system adheres to legal considerations regarding data privacy (Data Privacy Act of 2012) by ensuring guest information and receipt images are securely stored and accessible only to the administrator.
- **Maintainability:** The source code follows standard engineering practices (PEP 8, Component-based architecture) to ensure the software is easy to update or fix by future developers.
- **Schedule:** The project is strictly constrained by the academic calendar, requiring the final prototype to be fully operational and tested by April 2026.
- **Sustainability:** The design promotes sustainability by replacing the resort's paper-based logbooks and physical receipts with a digital database, effectively reducing paper waste.
- **Usability:** The user interface is designed for high usability, ensuring that resort staff with minimal technical training can navigate the dashboard and manage bookings efficiently.

1.7 Engineering Standards

To ensure the system's reliability, maintainability, and code quality, the project strictly adheres to the following industry-recognized engineering standards:

ISO/IEC 25010:2011 (Systems and Software Quality Models) This standard defines a product quality model which includes characteristics such as functional suitability, performance efficiency, and usability. The system shall follow the ISO/IEC 25010 standard to ensure that the booking interface is intuitively designed for non-technical users (Usability) and that the reservation logic prevents double-booking errors with 100% accuracy (Functional Suitability).

W3C Web Standards (HTML5 & CSS3 Architecture) This standard, established by the World Wide Web Consortium, defines the open standards for the structural and presentational layers of web applications. The system shall follow W3C standards for HTML and CSS to guarantee cross-platform compatibility, ensuring the resort's website renders correctly and responsively on both desktop browsers and mobile devices.

PEP 8 (Python Enhancement Proposal 8) This standard is the de facto style guide for Python code, providing conventions for writing readable and consistent code. The system shall follow PEP 8 guidelines for all backend development (Django) to ensure maintainability. This includes using 4 spaces for indentation, snake_case for function and variable names (e.g., calculate_total), and limiting line lengths to 79 characters to improve code readability for future developers.

ISO/IEC 27001:2022 (Information Security, Cybersecurity and Privacy Protection) This standard provides the framework for an Information Security Management System (ISMS) to preserve the confidentiality, integrity, and availability of information. The system shall follow basic ISO/IEC 27001 principles regarding Access Control to ensure that sensitive guest data (names, contact numbers) and financial proofs (receipts) are accessible only by the authorized Administrator.

IEEE 1016-2009 (Standard for Information Technology—Software Design Descriptions) This standard specifies the content and organization of software design descriptions (SDD) to support the design, testing, and maintenance of software systems. The system documentation shall follow IEEE 1016 guidelines to clearly define the software architecture, data flow (DFD), and database schema (ERD), ensuring the project structure is understandable for future maintenance or upgrades.

1.8 Engineering Design Process

The development of the "**Development of an Online Web-Based Booking System for Reservation and Resort Operations**" follows the iterative **Engineering Design Process** (EDP). This systematic approach ensures that the final software solution effectively solves the client's operational problems while adhering to the identified constraints.

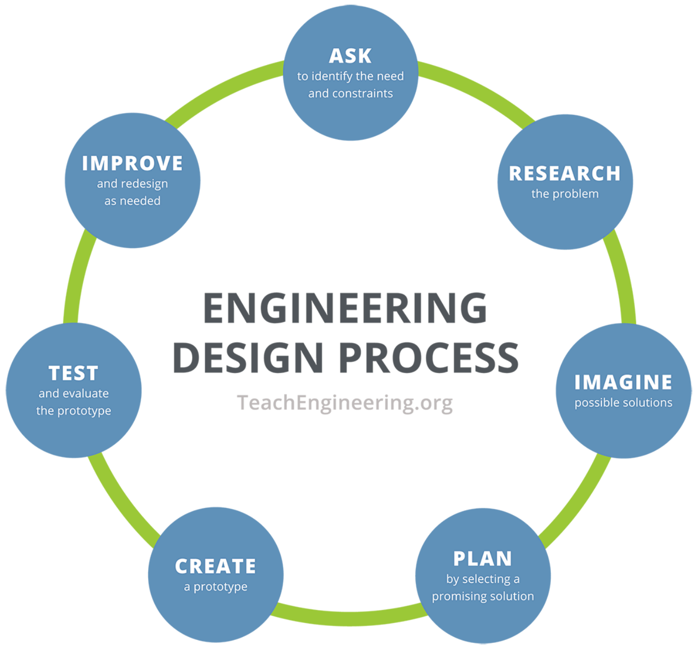
ASK
to identify the need and constraints
RESEARCH
the problem
IMAGINE
possible solutions
PLAN
by selecting a promising solution
CREATE
a prototype
TEST
and evaluate the prototype
IMPROVE
and redesign as needed
ENGINEERING DESIGN PROCESS
TeachEngineering.org

Figure 1.2 The Engineering Design Process (TeachEngineering, 2023)

As illustrated in **Figure 1.2**, the process is a continuous cycle rather than a linear path. The team moves through identifying the problem, researching solutions, planning the architecture, and building the prototype. The cycle allows for testing and redesigning (Improve phase) to refine the system based on feedback or error detection.

1.8.1 Ask: Identifying the Need and Constraints

The process began by consulting with the client, Ms. Christina Limin, to identify the core problem: the inefficiency of manual logbooks which leads to double-bookings and difficulty in retrieving guest records.

- Need: A centralized digital platform to automate reservations and showcase amenities.
- Constraints: The solution was bound by a zero-cost budget (requiring open-source tools), a strict academic deadline (April 2026), and the need for cross-platform accessibility (Web-based vs. Mobile App).

1.8.2 Research the Problem

The team analyzed the current manual workflow of the resort to understand the specific data points required (e.g., Guest Name, Date, Payment Receipt).

- Activity: The team reviewed existing booking platforms (such as Agoda or simple appointment systems) and researched the "Proof of Payment" verification method to replace expensive payment gateway integrations.
- Outcome: It was determined that a Web-Based Application is the most viable solution as it requires no installation for guests and works on both phones and desktops.

1.8.3 Imagine: Develop Possible Solution

The team brainstormed potential architectures to solve the problem:

1. Mobile App (Android/iOS): Good for user experience but hard to maintain and distribute.
2. Desktop Application: Secure but limits guest access (cannot book from home).
3. Web Application (Responsive): The selected solution. It offers universal accessibility and centralized data management.

1.8.4 Plan: Select a Promising Solution

The team selected the Python (Django) and HTML/CSS stack for its robust security and rapid development capabilities.

- Activity: The team created architectural blueprints, including the Data Flow Diagram (DFD) to map user interaction and the Entity Relationship Diagram (ERD) to structure the database.
- Activity: A project timeline (Gantt Chart) was established to ensure the prototype is ready for testing by the deadline.

1.8.5 Create: Build a Prototype

The development phase involved coding the "Minimum Viable Product" (MVP).

- Frontend: Developing the user interface (Home, Gallery, Booking Form) using HTML and CSS to ensure responsiveness.
- Backend: Configuring the Django server to handle date validation logic and prevent overlapping reservations.
- Database: Setting up the SQL database to store guest records and receipt images securely.

1.8.6 Test and Evaluate the Prototype

The prototype undergoes rigorous testing to verify it meets the Engineering Standards defined in Section 1.7.

- Functional Testing: Simulating simultaneous bookings to ensure the system successfully blocks the second attempt (preventing double-booking).

- Usability Testing: Presenting the Admin Dashboard to the client to ensure non-technical staff can easily approve or reject requests.

1.8.7 Improve: Redesign as Needed

Based on test results, the system is refined.

- **Iteration:** If testing reveals that the "Upload Receipt" process is too slow, the team optimizes the image compression algorithm.
- **Future Proofing:** While current features are locked for the academic deadline, the team documents potential improvements (such as automatic SMS gateways or AI chatbots) for future upgrades.

CHAPTER 2: SOFTWARE DESIGN

This chapter presents the technical foundation of the proposed Web-Based Booking System for Palacio Feliz – Event and Private Resort. It discusses the system architecture, engineering principles applied, prior art comparison, and the design alternatives considered before selecting the final solution.

2.1 Description of the Design Solution

2.1.1 General Description

The proposed solution is a centralized and responsive Web-Based Booking System developed for Palacio Feliz – Event and Private Resort. The system replaces the resort's manual logbook and social media-based reservation process with a structured digital platform designed to improve operational efficiency, scheduling reliability, and record organization.

The application operates using a client–server architecture. Guests access the system through a web interface to browse amenities, check room availability, and submit reservation requests with proof of payment. Resort management accesses an administrative dashboard to review requests, verify payments, and manage schedules.

The system centralizes booking records within a cloud-hosted database, allowing controlled administrative access and structured data storage. By transitioning from manual to digital operations, the platform enhances workflow organization, minimizes scheduling conflicts, and supports long-term operational scalability.

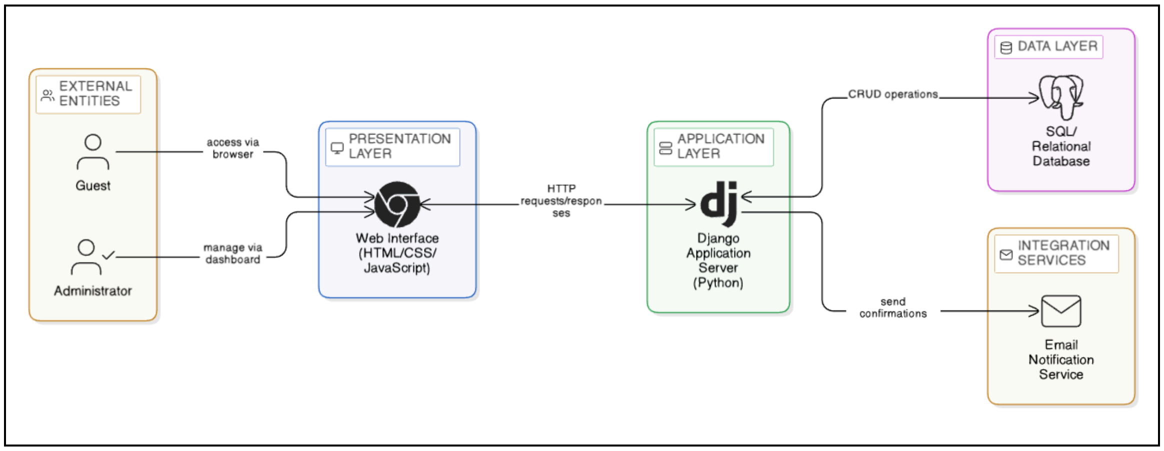

Figure 2.1 Client–Server Architecture of the Proposed Web-Based Booking System

Figure 2.1 illustrates the layered client–server architecture of the proposed Web-Based Booking System. The External Entities consist of the Guest and the Administrator, who access the system through a web interface using standard web browsers. The Presentation Layer handles user interaction through HTML5, CSS3, and JavaScript, enabling users to submit booking forms and manage reservations.

The Application Layer, implemented using the Django framework (Python), processes HTTP requests and responses, performs booking validation, manages authentication, and executes business logic. This layer ensures that selected dates are checked against existing reservations to prevent double-bookings.

The Data Layer consists of a relational database that stores guest information, reservation schedules, payment records, and administrator credentials. Additionally, the system integrates with an Email Notification Service to send booking confirmations and status updates to users.

This architecture ensures centralized control, real-time scheduling validation, data integrity, and secure management of resort operations.

2.1.2 Engineering Principles Involved

To ensure that the proposed Web-Based Booking System satisfies the constraints of functionality, usability, maintainability, and data accuracy, the design incorporates the following core engineering principles:

1. Model-View-Template (MVT) Architecture

- **Principle:** MVT is a software design pattern used by the Django framework. It separates the application into three distinct components: the *Model* (database structure and data handling), the *View* (request processing and business logic), and the *Template* (user interface presentation).
- **Relevance:** By decoupling the user interface from the data logic, the system ensures that changes to the visual design (e.g., updating room photos) do not risk breaking the reservation logic. This directly supports the maintainability constraint required for future upgrades.

2. Client–Server Architecture

The system implements a Client–Server model, where user requests originate from client devices (via web browsers) and are processed centrally by the Django application server. This architecture ensures:

- Centralized data storage
- Real-time validation of booking dates
- Controlled administrator access

By centralizing booking logic on the server side, the system prevents users from bypassing validation rules, ensuring zero double-bookings. This principle directly addresses the Functionality constraint (Booking Error Rate).

3. Server-Side Validation and State Management

All reservation requests undergo server-side validation before being committed to the database. The system compares selected booking dates against existing records to prevent overlapping reservations.

Additionally, the booking process follows a controlled state transition model:

NULL → PENDING → RESERVED (or CANCELLED)

This state management approach ensures that bookings are not finalized until administrative verification is completed. This improves data accuracy and operational reliability.

4. Responsive Web Design (RWD)

The system uses responsive web design techniques such as flexible layouts and CSS media queries to ensure usability across multiple devices, including desktops, tablets, and smartphones.

This principle ensures compliance with the Accessibility and Usability constraints by maintaining consistent user experience across different screen sizes.

5. Modular Design and Separation of Concerns

Each system module (Booking, Authentication, Admin Dashboard, Email Notification) is implemented independently. This modular structure reduces complexity and allows future developers to modify or extend features without affecting the entire system.

This principle enhances long-term sustainability and maintainability of the system.

Supporting Engineering Literature

The engineering principles applied in this system are supported by established software engineering literature. The Model–View–Template (MVT) architectural pattern is widely recognized for promoting separation of concerns, improving maintainability, and reducing system complexity in web-based applications (Gamma et al., 1994). Client–Server architecture is a standard distributed computing model that centralizes processing and data management, enhancing scalability and control (Tanenbaum & Van Steen, 2007).

Server-side validation is recommended in secure web system development to prevent unauthorized data manipulation and ensure data integrity (OWASP, 2021). Additionally, responsive web design principles ensure cross-platform accessibility and usability, which are essential characteristics of high-quality software systems as defined by ISO/IEC 25010:2011.

These established principles provide a theoretical and engineering foundation for the design decisions implemented in the proposed Web-Based Booking System.

2.1.3 Prior Art Analysis

To evaluate the relevance and originality of the proposed Web-Based Booking System, existing reservation management solutions were analyzed. These include traditional manual logbook systems and Online Travel Agency (OTA) platforms such as Agoda and Airbnb.

Manual reservation systems rely on physical logbooks and social media messaging. While simple and low-cost, this method lacks real-time validation, is prone to double-booking errors, and exposes records to potential loss or damage.

On the other hand, OTA platforms provide automated booking management, online payment integration, and global exposure. However, these systems charge service fees, limit customization, and do not provide full data ownership to the resort.

The proposed system bridges the gap by offering automation and real-time availability checking while maintaining full branding control and zero operational cost.

Prior Art Analysis Matrix

Design	**Features**		
	Manual Logbook (Current)	**OTA Platforms (e.g., Agoda, Airbnb))**	**Proposed Web System**
Real-time Availability	**X**	✔	✔
Double-Bookin	**X**	✔	✔

Design	Features		
	Manual Logbook (Current)	OTA Platforms (e.g., Agoda, Airbnb))	Proposed Web System
g Prevention			
Custom Branding	✔	X	✔
No Third-Party Platform Fees	✔	X	✔
Full Data Ownership	✔	X	✔
Admin Dashboard Control	X	Limited	✔
Payment Proof Upload	Manual	Integrated Gateway	Screenshot Upload

Table 2.1 Prior Art Analysis Matrix

Table 2.1 presents a comparative analysis between the existing manual logbook system, OTA platforms, and the proposed web-based system. While OTA platforms provide automation and integrated payment processing, they impose commission fees and limit branding control. The manual logbook offers full ownership and zero platform dependency but lacks automation and real-time validation. The proposed system integrates automation, centralized scheduling, full data ownership, and administrative control without relying on third-party platforms. This differentiation highlights the originality and practical relevance of the proposed design.

2.2 General System Architecture

This section describes the structural organization of the Web-Based Booking System and the interaction between its major components. The system is implemented using a layered client–server architecture consisting of external users, a presentation layer, an application layer, and a centralized relational database. Each layer performs a distinct responsibility to ensure separation of concerns and controlled data flow.

2.2.1 Hardware Elements

Server Host:
A virtualized cloud server (PythonAnywhere) is used to host the Django application and database. This eliminates the need for on-premises physical infrastructure and ensures 24/7 accessibility.

Client Terminals:
The system is hardware-agnostic and can be accessed using smartphones (Android/iOS), tablets, laptops, or desktop computers capable of running modern web browsers.

2.2.2 Software Elements

A. Embedded Software

Not applicable. The proposed system does not rely on embedded firmware or hardware-level programming components, as it operates entirely through web technologies.

B. Application Software

- **Backend:** Python 3.x using the **Django Framework**. Django handles URL routing, session management, and database queries.
- **Frontend: HTML5** for structure, **CSS3** for styling (Responsive Design), and **JavaScript** for dynamic DOM manipulation (e.g., calendar updates).
- **SQLite:** Used for the development and prototyping phase due to its serverless configuration and easy integration with Python, storing Guest Profiles and Reservation Records.

C. Key Algorithms Used

The system implements a booking validation algorithm that checks for date conflicts before confirming reservations. The algorithm ensures that selected dates do not overlap with existing approved bookings, thereby preventing double-booking errors.

Additionally, authentication validation logic is implemented to verify administrator credentials before granting dashboard access.

2.2.3 Data, Datasets, and Processing

a. Datasets

The system processes three primary datasets acquired directly from user input and administrator configuration:

1. **Guest Data:**
   Collected through the reservation form. This includes Full Name, Contact Number, Email Address, and optional Facebook Profile Link.
2. **Booking Data:**
   Generated during the reservation transaction. This includes Check-in Date, Check-out Date, Selected Room Type, Reservation Status, and uploaded Proof of Payment.
3. **Resort Data:**
   Managed by the Administrator. This includes Room Names, Descriptions, Pricing Constants, Schedule Availability Flags, and stored reservation calendar records.

b. Data Processing Scheme and Algorithms

The system processes booking transactions through a structured Input–Process–Output (IPO) model. Each stage incorporates validation controls to ensure consistency between user-submitted data and stored reservation records.

Pre-Processing (Input Sanitization):

Upon form submission, Django's built-in validation mechanisms sanitize text inputs to prevent malicious script injection (XSS attacks). Uploaded files are validated to ensure only permitted image formats (JPG/PNG) are accepted.

Processing (Logic Application):

- **Availability Validation Algorithm:**
  The system compares the selected reservation dates against existing confirmed reservations stored in the database. Overlapping date ranges result in rejection or pending status.

- **State Transition Control:**
  The booking status follows a controlled state sequence:

  NULL → PENDING → RESERVED or CANCELLED.

This state-machine structure ensures consistent booking lifecycle management.

Post-Processing (Output Generation):

When a booking is approved, the system generates and sends a templated confirmation message to the guest's registered email address. The reservation calendar is automatically updated to reflect the confirmed dates.

c. Other Data Utilized in the Design

Mock Data for Testing:

During system evaluation, a dataset of 50 simulated reservations was used to stress-test date overlap detection and calendar integrity.

Session Data:

The system utilizes secure session IDs and browser cookies to maintain authenticated administrator sessions while preventing unauthorized dashboard access.

2.3 Design Alternatives

This section discusses alternative architectural and implementation approaches considered during system development and justifies the final design selection.

2.3.1 Rationale for Design Alternatives

The primary constraints of this project include a zero-cost development budget and a fixed academic deadline. Given that the backend technology stack (Python/Django) was predetermined, the design alternatives focused primarily on frontend architecture and user navigation flow.

These alternatives were evaluated based on the following criteria:

- Functionality
- Cost
- Schedule Feasibility
- Usability
- Maintainability

2.3.2 Design Alternative A: Single-Page Application (AJAX-Based)

A. Engineering Principle

This design is based on Asynchronous JavaScript (AJAX), where content dynamically updates without full page reloads. The system operates within a single HTML shell while switching views programmatically.

B. Architecture

The architecture consists of a single-page layout (index.html) where different views (Home, Booking, Rooms) are shown or hidden using JavaScript.

C. Constraint Evaluation

- **Functionality:** Moderate Risk. Handling booking validation and server-side error states without page reloads increases the possibility of state inconsistencies.
- **Cost:** Low. Utilizes open-source technologies.
- **Schedule:** High Risk. Requires advanced JavaScript frameworks (e.g., React or Vue.js), increasing development complexity and learning curve.
- **Usability**: Moderate. Provides modern interaction but may reduce compatibility on lower-performance devices.

Conclusion: The hierarchical multi-page architecture provides a balanced integration of structured navigation, server-controlled validation, and modular organization. Compared to the other alternatives, this design achieves optimal alignment with functionality, usability, schedule feasibility, and maintainability constraints.

2.3.3 Design Alternative B: Linear Wizard Process (Tunneling)

A. Engineering Principle

This design follows the Sequential Processing Principle, enforcing a strict step-by-step workflow.

B. Architecture

User flow is fixed:

Select Date → Select Room → Upload Receipt → Confirm.

The site contains minimal navigation outside the booking tunnel.

C. Constraint Evaluation

- **Functionality:** High. Reduces error probability and prevents skipped steps.
- **Cost:** Low. No additional tools required.
- **Schedule:** Feasible. Simple logical structure.
- **Usability:** Low. Restricts exploratory browsing, limiting user flexibility and potentially reducing user satisfaction.

Conclusion: Although highly controlled and error-resistant, this design limits user freedom and negatively affects browsing experience.

2.3.4 Design Alternative C: Hierarchical Multi-Page Architecture

A. Engineering Principle

This design follows Django's Model-View-Template (MVT) architecture using Server-Side Rendering (SSR). Each page is independently rendered and managed by the server.

B. Architecture

The website consists of distinct semantic pages (Home, Gallery, Services, Book Now) connected through a persistent navigation bar.

C. Constraint Evaluation

- **Functionality:** High. Booking validation is isolated to a dedicated page, minimizing logical conflicts.

- **Cost:** Low. Uses open-source technologies aligned with project constraints.
- **Schedule:** Excellent. Aligns directly with Django's standard documentation and development workflow.
- **Usability:** High. Mirrors conventional travel website structures, enhancing user familiarity.
- **Maintainability:** High. Modular page structure simplifies debugging and future modifications.

Conclusion: The hierarchical multi-page architecture best satisfies the project's constraints while balancing functionality, usability, and maintainability.

2.4 Standards Involved in the Design

This section presents the standards followed by the design, including their references. Matrix may be used to show how standards are used in each specific design.

Table 2.2 Summary of Standards Involved in the Alternatives

Standard	Brief Description	DESIGNS		
		DESIGN A (AJAX-Based)	DESIGN B(Linear Wizard)	DESIGN C(Multi-Page Architecture)
ISO/IEC 25010:2011	Defines a product quality model covering functional suitability, performance efficiency, and usability	Used to validate the system's ability to prevent double-bookings and ensure dashboard usability.		
W3C Web Standards	Defines open standards for the structural and presentational layers of web applications using HTML5 and CSS3.	Applied to frontend development to guarantee cross-platform compatibility on desktops and mobile devices		
ISO/IEC 27001:2022	Provides a framework for an Information Security Management System (ISMS) to preserve data	Implemented to ensure access control so sensitive guest data and payment receipts	Implemented to ensure access control so sensitive guest data and payment receipts	Implemented to ensure access control so sensitive guest data and payment receipts

	confidentiality and integrity	are strictly for authorized Administrator access.	are strictly for authorized Administrator access	are strictly for authorized Administrator access.
PEP 8	The industry-standard style guide for Python code, providing conventions for indentation, naming, and structure.	Applied to the predetermined Django backend code to ensure maintainability and readability	Applied to the predetermined Django backend code to ensure maintainability and readability	Applied to the predetermined Django backend code to ensure maintainability and readability
IEEE 1016-2009	Specifies the content and organization of software design descriptions (SDD).	Guides the documentation of software architecture, database schema (ERD), and data flow (DFD)	Guides the documentation of software architecture, database schema (ERD), and data flow (DFD).	Guides the documentation of software architecture, database schema (ERD), and data flow (DFD)

- The ISO/IEC 25010:2011 standard ensures that the booking interface remains highly intuitive for non-technical staff and maintains 100% accuracy in preventing overlapping schedules.

- The W3C Web Standards guarantee that the user interface renders correctly and responsively across various mobile and desktop browsers.

- To secure user information, ISO/IEC 27001:2022 protocols are applied to restrict access to sensitive guest records and financial proofs exclusively to the authorized Administrator.

- For the backend logic, PEP 8 guidelines dictate Python coding conventions—such as using 4 spaces for indentation and snake_case for variables to maximize code readability.

- Finally, IEEE 1016-2009 informs the project's documentation structure, standardizing the presentation of the system's architecture, data flows, and database schemas.

CHAPTER 3: DESIGN TRADEOFFS

3.1 Summary of Constraints

Explain table xx below in this paragraph.

Table xx Summary of Design Constraints

Designs	Constraints				
	Constraint A (Metric)	Constraint B (Metric)	Constraint C (Metric)	Constraint D (Metric)	Constraint E (Metric)
Design A					
Design B					
Design C					

Synthesize for the next section.

3.2 Trade-offs

Table xx Preference and Importance of Constraints

Constraints	Preference	Importance (raw)	% Importance

Explain the use of Pareto Multi-Criteria Decision Making (MCDM).

$$Minimization = 9 \times \left(\frac{Max\,Value - Raw\,Value}{Max\,Value - Min\,Value} \right) + 1 \text{ Equation No. xx}$$

$$Maximization = 9 \times \left(\frac{Raw\,Value - Min\,Value}{Max\,Value - Min\,Value} \right) + 1 \text{ Equation No. xx}$$

3.2.1 Tradeoff 1: Constraint A (Metric)

3.2.1.1 Design 1: Normalization of Constraint A (Metric)
<Introduce>

Table xx Evaluation of Three Design Alternatives based on Constraint A

Design	**Constraint (Metric)**

<Analyze>

3.2.1.2 Design 2: Normalization of Constraint A (Metric)

Table xx Evaluation of Three Design Alternatives based on Constraint B

Design	**Constraint (Metric)**

3.2.1.3 Design 3: Normalization of Constraint A (Metric)

Table xx Evaluation of Three Design Alternatives based on Constraint C

Design	**Constraint (Metric)**

3.2.2 Tradeoff 2: Constraint B (Metric)

3.2.2.1 Design 1: Normalization of Constraint B (Metric)
3.2.2.2 Design 2: Normalization of Constraint B (Metric)
3.2.2.3 Design 3: Normalization of Constraint B (Metric)

3.2.3 Tradeoff 3: Constraint C (Metric)

3.2.3.1 Design 1: Normalization of Constraint C (Metric)
3.2.3.2 Design 2: Normalization of Constraint C (Metric)
3.2.3.3 Design 3: Normalization of Constraint C (Metric)

3.2.4 Tradeoff 4: Constraint D (Metric)

3.2.4.1 Design 1: Normalization of Constraint D (Metric)
3.2.4.2 Design 2: Normalization of Constraint D (Metric)
3.2.4.3 Design 3: Normalization of Constraint D (Metric)

3.2.5 Tradeoff 5: Constraint E (Metric)

3.2.5.1 Design 1: Normalization of Constraint E (Metric)
3.2.5.2 Design 2: Normalization of Constraint E (Metric)
3.2.5.3 Design 3: Normalization of Constraint E (Metric)

3.3 Summary of the Normalized Values of the Three Designs

Designs	**Constraints**				
	Constraint A (metric)	**Constraint B (metric)**	**Constraint C (metric)**	**Constraint D (metric)**	**Constraint E (metric)**
Design A					
Design B					
Design C					

3.4 Designers Raw Ranking for the Three Designs

Table xx Designers Raw Ranking for the Three Designs

Decision Criteria	**Criterion's Importance**		**Ability to Satisfy Criterion**		
	Scale (0-10)	**Percentage (%)**	**Design A**	**Design B**	**Design C**

3.5 Sensitivity Analysis

3.6 Influence of the Design Tradeoffs in the Final Design

CHAPTER 4: FINAL DESIGN

4.1 Final Design

4.1.1 Software Application

4.1.2 Hardware Design

4.2 Test Procedures and Evaluation

4.2.1 Test Procedures

4.2.2 Test Evaluation

4.3 Test and Evaluation Results

4.3.1 Test Results

4.3.2 Evaluation Results

4.4 Conclusion

4.5 Impact of the Design

4.5.1 Societal

Target UN SDG.

4.5.2 Ethical

In compliance with known ethical codes/standards

4.5.3 Legal

National / Intl Laws

4.6 Sustainability Plan

CHAPTER 5: BUSINESS PLAN AND MODEL

REFERENCES

Note: This must be done using APA format. Check the guide for more details: https://www.scribbr.com/apa-style/apa-seventh-edition-changes/

Covey, S. R. (2013). *The 7 habits of highly effective people: Powerful lessons in personal change*. Simon & Schuster.

APPENDICES

Include standards preview, certification from experts/clients, code snippets, patent reports, and other long and detailed documents here. Format is as follows below:

APPENDIX A: TITLE OF THE SECTION

<figure>

Note: No figure number. Standards must be followed with a paragraph explaining its contents and purpose.