horizontal line

Software Development Plan

**Github link**:https://github.com/MAbaset/Five-hotlels-

**April 7, 2024**

Version 1.0

**Submitted By:**

**Mohamed Abd Elbaset**

**Mahmoud Mohamed Sharfy**

**Mostafa Mahmoud Elsayed**

**Ahmed Mohamed Nazeer**

[**PRODUCT DESCRIPTION**](#_7mtis1co57cx) **3**

# The product is a modern hotel website designed to streamline booking processes, enhance user experience, and improve customer engagement. The client is a boutique hotel chain looking to expand its online presence and attract more guests. The problem being solved is the need for a user-friendly and visually appealing platform that facilitates seamless reservations and provides comprehensive information about the hotel's amenities and services.

# 2. TEAM DESCRIPTION

|  |  |  |
| --- | --- | --- |
| Role | Skill required | Team members |
| Web Developer | HTML, CSS, JavaScript, PHP/Python/, CMS experience | Mohamed Abd Elbaset |
| Ul/UX Designer | Adobe XD/Sketch/Figma, User research, Wireframing | Mahmoud Mohamed Sharfy |
| Backend Developer | Nodejs, Database management (SQL/NoSQL), API integration | Mostafa Mahmoud Elsayed  Ahmed Mohamed Nazeer |
|  |  |  |

# **3. SOFTWARE PROCESS MODEL DESCRIPTION**

# 1. PRODUCT DESCRIPTION

# For a simple application like a hotel reservation system where requirements are relatively clear and unlikely to change significantly, an Agile approach may be overly complex and unnecessary. Instead, a waterfall model could be more suitable

Waterfall Model:

**1- Requirements Gathering**

Gather all requirements for the hotel reservation system upfront, including user registration, room selection, and reservation functionalit

**2- Design**

Design the system architecture, database schema, and user interface based on the gathered requirements.

**3- Implementation**

Develop the system according to the design specifications. This involves coding the registration, room selection, and reservation features.

**4- Testing**

Conduct comprehensive testing to ensure that all components of the system function correctly and meet the specified requirements.

**5- Deployment**

Deploy the system for users to access and interact with.

**6- Maintenance**

Provide ongoing support and maintenance to address any issues or bugs that arise post-deployment.

# **4. PRODUCT DEFINITION**

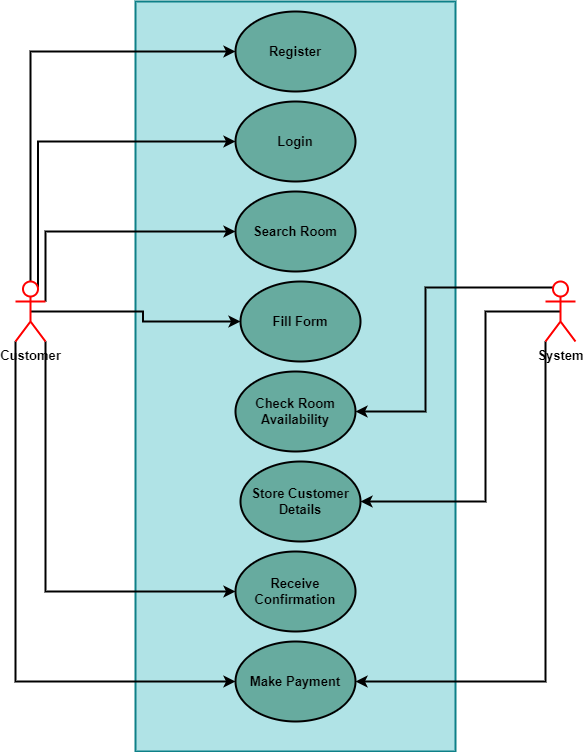
## **Context diagram**

## 

**Customer:**

**The Customers are the everyday record collectors/appreciators. They can create an account with a username and password and they need to input an email. These users will create a running wishlist of specific records they are looking for or want to purchase. The user can also look up the hotel they want and see if any local hotel have it in their system. If a user is looking for a specific hotel, they can also look up them too. The user can like the hotel or the room to be notified when certain events happen.**

## [**Use Cases**](https://docs.google.com/document/d/12XgfNrVNvVZUDHk7BYBtv-j7iTASqcnU_EvzfFDVBv4/edit) **(click for individual descriptions)**



Use Case #1:

Name: Register

Participating Actor(s): Customers

Entry Conditions: User wants to create a new account

Exit Conditions: User successfully registers and receives confirmation

Flow of Events:

1. User navigates to the registration page.
2. User fills in required information such as name, email, password, etc.
3. User submits the registration form.
4. System validates the information provided by the user.
5. If the information is valid, the system creates a new account for the user.
6. User receives a confirmation message/email.
7. User can now log in using the newly created credentials.

Special Requirements: Ensure secure handling of user data, such as encryption of passwords and protection against SQL injection.

Use Case #2:

Name: Login

Participating Actor(s): Customers

Entry Conditions: User has an existing account

Exit Conditions: User successfully logs in and gains access to their account

Flow of Events:

* User navigates to the login page.
* User enters their username/email and password.
* System validates the login credentials.
* If the credentials are valid, the system grants access to the user's account.
* User is redirected to their account dashboard or home page.
* If the credentials are invalid, an error message is displayed, prompting the user to try again.

Special Requirements: Implement security measures such as rate limiting for login attempts and CAPTCHA to prevent brute-force attacks.

Use Case #3:

Name: Search Room

Participating Actor(s): Customers

Entry Conditions: User is logged in

Exit Conditions: User views available rooms matching their search criteria

Flow of Events:

* User navigates to the room search page.
* User specifies search criteria such as check-in/out dates, number of guests, room type, etc.
* System retrieves available rooms based on the specified criteria.
* User views the list of available rooms.
* User can filter or sort the list based on preferences.
* User selects a room they are interested in for further details or booking.

Special Requirements: Ensure real-time updates on room availability to avoid double bookings.

Use Case #4:

Name: Fill Form

Participating Actor(s): Customers

Entry Conditions: User has selected a room for booking

Exit Conditions: User completes the booking form

Flow of Events:

1. User selects a room they wish to book.
2. User is prompted to fill in the booking form.
3. User provides necessary information such as guest details, special requests, etc.
4. User submits the booking form.
5. System validates the provided information.
6. If the information is valid, the booking process continues.
7. If the information is invalid, an error message is displayed, prompting the user to correct the errors.

Special Requirements: Ensure user-friendly interface for easy form filling and validation.

Use Case #5:

Name: Check Room Availability

Participating Actor(s): System

Entry Conditions: User selects check-in/out dates and room type

Exit Conditions: System displays available rooms for the specified dates and room type

Flow of Events:

* System receives the check-in/out dates and room type selected by the user.
* System checks the availability of rooms for the specified dates and room type.
* System retrieves a list of available rooms.
* System displays the list of available rooms to the user.

Special Requirements: Implement efficient algorithms to handle large volumes of data for room availability checks.

Use Case #6:

Name: Store Customer Details

Participating Actor(s): System

Entry Conditions: User successfully fills in the booking form

Exit Conditions: Customer details are stored securely in the system

Flow of Events:

1. System receives the filled booking form from the user.
2. System validates the provided information.
3. If the information is valid, the system stores the customer details securely.
4. If the information is invalid, an error message is displayed, prompting the user to correct the errors.

Special Requirements: Implement data encryption and access controls to protect customer information.

Use Case #7:

Name: Receive Confirmation

Participating Actor(s): Customers

Entry Conditions: User successfully completes the booking process

Exit Conditions: User receives confirmation of their booking

Flow of Events:

1. User completes the booking process.
2. System generates a booking confirmation.
3. System sends the booking confirmation to the user via email or SMS.
4. User receives the booking confirmation.

Special Requirements: Ensure timely delivery of booking confirmations to users.

Use Case #8:

Name: Make Payment

Participating Actor(s): Customers

Entry Conditions: User has selected a room for booking

Exit Conditions: Payment is successfully processed

Flow of Events:

1. User selects a room they wish to book.
2. User fills in the booking form and confirms the booking details.
3. User selects a payment method (credit card, PayPal, etc.).
4. User provides payment information.
5. System processes the payment.
6. If the payment is successful, the booking is confirmed.
7. If the payment fails, an error message is displayed, prompting the user to try again or use a different payment method.

**Special Requirements: Implement secure payment gateway integration and adhere to PCI DSS standards for handling payment information.**

**5. PROJECT ORGANIZATION**

## **Matrix of Responsibilities**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Team Members  Concepts | Mohamed Abd Elbaset | Mostafa Mahmoud  Elsayed | Ahmed Mohmaed Nazer | Mahmoud Mohamed Sharfy |
| Database Management | X | X |  | x |
| Front-end |  | X | x | X |
| Back-end : PHP | X |  | X | X |
| UI/UX | x | X | X |  |

# [**PERT Chart**](https://drive.google.com/open?id=1rj1XyvQ63gfzoeDCI-cp3nwjir7VcBzN)



**7. VALIDATION PLAN**

## **Test Strategy**

Definition of Done:

The "definition of done" refers to the criteria or conditions that must be met for a task, feature, or project to be considered complete and ready for deployment. In the context of testing, the definition of done outlines the specific criteria that indicate when testing activities are finished and the website is ready to be launched. This typically includes:

1. All Test Cases Executed: Ensure that all planned test cases have been executed, including functional, usability, compatibility, security, and performance tests.

2. Bug Resolution: Address and resolve all identified bugs and issues, either by fixing them or providing acceptable workarounds.

3. Regression Testing: Conduct regression testing to verify that fixes or changes haven't introduced any new defects or impacted existing functionality.

4. User Acceptance Testing (UAT): Obtain approval from stakeholders or end-users through UAT, ensuring that the website meets their expectations and requirements.

5. Documentation: Ensure that all testing activities, results, and findings are properly documented for future reference and audit purposes.

6. Sign-off from Key Stakeholders: Obtain sign-off from key stakeholders, such as project managers, developers, and business owners, confirming that the website is ready for deployment.

Success Criteria:

The success criteria define what constitutes a successful outcome for the testing process. These criteria are closely related to the project objectives and may include both quantitative and qualitative measures of success. In the context of testing a hotel website, success criteria may include:

1. Functional Accuracy: Ensure that all website functionalities work as expected, without errors or unexpected behavior.

2. Usability and User Experience: Provide an intuitive and seamless user experience, allowing visitors to easily navigate the website and complete tasks such as booking rooms or accessing information.

3. Compatibility: Ensure that the website is compatible with a variety of devices (desktop, tablet, mobile) and web browsers (Chrome, Firefox, Safari, Edge) to reach a broader audience.

4. Security: Verify that the website is secure and protected against common security threats, such as data breaches or unauthorized access.

5. Performance: Ensure that the website loads quickly and responds promptly to user interactions, even under peak traffic conditions.

6. Stakeholder Satisfaction: Obtain positive feedback and approval from stakeholders, indicating their satisfaction with the website's functionality, design, and performance.

By clearly defining the "definition of done" and establishing success criteria, you can ensure that the testing process is thorough and effective, ultimately leading to a successful launch of the hotel website.

# **8. RISK ASSESSMENT**

**Risk Identification:**

1. **Technical Risks:**

- Lack of compatibility with certain browsers or devices.

- Performance issues due to high traffic or inadequate server capacity.

- Security vulnerabilities leading to data breaches or unauthorized access.

2. **Resource Risks:**

- Shortage of skilled developers or designers.

- Delays in obtaining necessary hardware or software resources.

- Dependence on third-party services for critical functionalities (e.g., booking system, payment gateway).

3. **Schedule Risks:**

- Unrealistic timelines leading to rushed development and testing.

- Dependencies on external factors (e.g., domain registration, content creation) causing delays.

4. **Scope Creep**:

- Changes in requirements or additional features requested by stakeholders.

- Unclear or evolving project scope leading to frequent adjustments.

**Risk Prioritization:**

Based on the identified risks, prioritize them according to their potential impact and likelihood of occurrence:

1. **High Priority Risks:**

- Security vulnerabilities

- Compatibility issues with major browsers/devices

- Resource shortages

- Schedule delays due to critical dependencies

2. **Medium Priority Risks:**

- Performance issues under high traffic

- Scope creep

- Technical challenges with integrations

3. **Low Priority Risks:**

- Minor compatibility issues

- Schedule delays due to non-critical dependencies

**Prioritized List:**

1. Security vulnerabilities

2. Compatibility issues with major browsers/devices

3. Resource shortages

4. Schedule delays due to critical dependencies

5. Performance issues under high traffic

6. Scope creep

7. Technical challenges with integrations

8. Minor compatibility issues

9. Schedule delays due to non-critical dependencies

**Risk Mitigation:**

1. Security Vulnerabilities:

- Conduct regular security audits and penetration testing.

- Implement security best practices such as data encryption and secure coding techniques.

2. Compatibility Issues:

- Test the website thoroughly across multiple browsers and devices.

- Use responsive design principles to ensure compatibility.

3. Resource Shortages:

- Hire additional resources or freelancers as needed.

- Cross-train team members to handle multiple roles if possible.

4. Schedule Delays:

- Break down tasks into smaller, manageable chunks with realistic timelines.

- Identify critical dependencies early and address them proactively.

**Addressing Risk Factors:**

- Security vulnerabilities: Continuous monitoring and updates, regular security audits.

- Compatibility issues: Comprehensive testing across browsers and devices, responsive design.

- Resource shortages: Hiring additional resources, cross-training team members.

- Schedule delays: Detailed project planning, proactive identification of critical dependencies.

- Performance issues: Optimization techniques, load testing.

- Scope creep: Clear definition of project scope, change control processes.

# **9. CONFIGURATION AND VERSION CONTROL**

Example: First major release, third minor feature update, second bug fix :: Version 1.5.2

# **10. TOOLS**

# **Development Tools:**

# Code Editor (e.g., Visual Studio Code, Sublime Text): Used for writing and editing HTML, CSS, JavaScript, and backend code.

# Version Control System (e.g., Git, GitHub): Facilitates collaboration among team members, tracks changes, and manages code versions.

# Local Development Server (e.g., XAMPP, WampServer): Allows testing of website functionalities locally before deployment.

# **Design Tools:**

# Adobe XD, Sketch, Figma: Used for creating wireframes, mockups, and designing the user interface (UI) and user experience (UX) of the website.

# Adobe Photoshop, Illustrator: For editing images, creating logos, and graphics used on the website.

# Backend Development Tools:

# Node.js: For backend development and server-side scripting.

# Database Management System (e.g., MySQL, PostgreSQL, MongoDB): Stores and manages data related to bookings, user profiles, etc.

# API Integration Tools (e.g., Postman): Used for testing and integrating APIs for payment gateways, booking systems, etc.

# Digital Marketing and Analytics Tools:

# Google Analytics: Tracks website traffic, user behavior, and provides insights for digital marketing strategies.

# SEO Tools (e.g., SEMrush, Moz): Assists in keyword research, on-page SEO optimization, and monitoring search engine rankings.

# Email Marketing Platform (e.g., MailChimp, Constant Contact): Manages email campaigns, newsletters, and customer communication.

# Security Tools

# SSL Certificate: Ensures secure data transmission by encrypting user information during online transactions.

# Firewall and Security Plugins: Protects the website from potential threats, such as malware, hacking attempts, etc.

# Customer Support Tools:

# CRM (Customer Relationship Management) System (e.g., Salesforce, HubSpot): Manages customer interactions, inquiries, and support tickets.

# Live Chat Software (e.g., Intercom, LiveChat): Enables real-time communication with website visitors for instant support.

# These tools collectively support the development, design, marketing, security, and customer support aspects of the hotel website project. Depending on specific requirements and preferences, variations or additional tools may be used.

# **11. ARCHITECTURE**

* Devices for testing
  + Android
  + iPhone
* Computers for programming