Hotel Web Application

Software Design Document  
Andrew D. Pitt

Contents

[1. INTRODUCTION 2](#_Toc505770981)

[1.1  Purpose 2](#_Toc505770982)

[1.2  Scope 2](#_Toc505770983)

[1.3  Overview 2](#_Toc505770984)

[1.4  Reference Material 3](#_Toc505770985)

[2. SYSTEM OVERVIEW 3](#_Toc505770987)

[3. SYSTEM ARCHITECTURE 3](#_Toc505770988)

[3.1  Architectural Design 3](#_Toc505770989)

[3.2  Decomposition Description 4](#_Toc505770990)

[3.3  Design Rationale 4](#_Toc505770991)

[4. DATA DESIGN 4](#_Toc505770992)

[4.1  Data Description 4](#_Toc505770993)

[4.2  Data Dictionary 5](#_Toc505770994)

[5. COMPONENT DESIGN 6](#_Toc505770995)

[6. HUMAN INTERFACE DESIGN 8](#_Toc505770996)

[6.1  Overview of User Interface 8](#_Toc505770997)

[6.2  Screen Objects and Actions 9](#_Toc505770998)

# 

# 1. INTRODUCTION

## 1.1  Purpose

This software design document describes the architecture and system design of a web application for a hotel in order to give development, security, and administrative teams guidance in composition of the system. It will meet user flows outlined below in section 1.2, as well as the security concerns of the hotel in its interest to protect assets. The SDD covers the high level system design as well the low-level details of component architecture, behavior, and protection.

## 1.2  Scope

This document is for a web application which must have a public-facing interface for customer use, as well as an elevated privilege user interface for staff to manage data within the system. Adequate security measures are required in order to protect assets in places that that leave the system open to ongoing threat. There are a few interoperable components (server, database, and other) within the back end of the system, and each one of them has a unique set of vulnerabilities based on its role and physical location/setup within the system. In development of this web application, we must be mindful of the user flow requirements in order to meet the hotel’s goals alongside those security concerns. The web app is intended to provide efficient and confidential ways for customers and the business to interact, and also to provide large-scale availability of service.   
The staff must be able to:

* See room availability
* Check in and check out customers
* View and modify inventory
* Open maintenance tickets and track progress

In addition, customers must be able to:

* See room availability
* Reserve a room

## 1.3  Overview

This document is broken into sections which gradually deepen into the details of the architecture and the data managed by it.   
Section 1 is intended to provide background information and identify goals.  
Section 2 is intended to provide a brief description of functionality, context, and design of the web application.   
Section 3 provides an outline of the architecture using description and diagrams.  
Section 4 outlines the data structures and flow within the web application.   
Section 5 outlines the alignment of security goals with user goals, as dictated by user flows.  
Section 6 describes the human interface design; specifically the functionality of hte system from the user’s perspective.

## 1.4  Reference Material

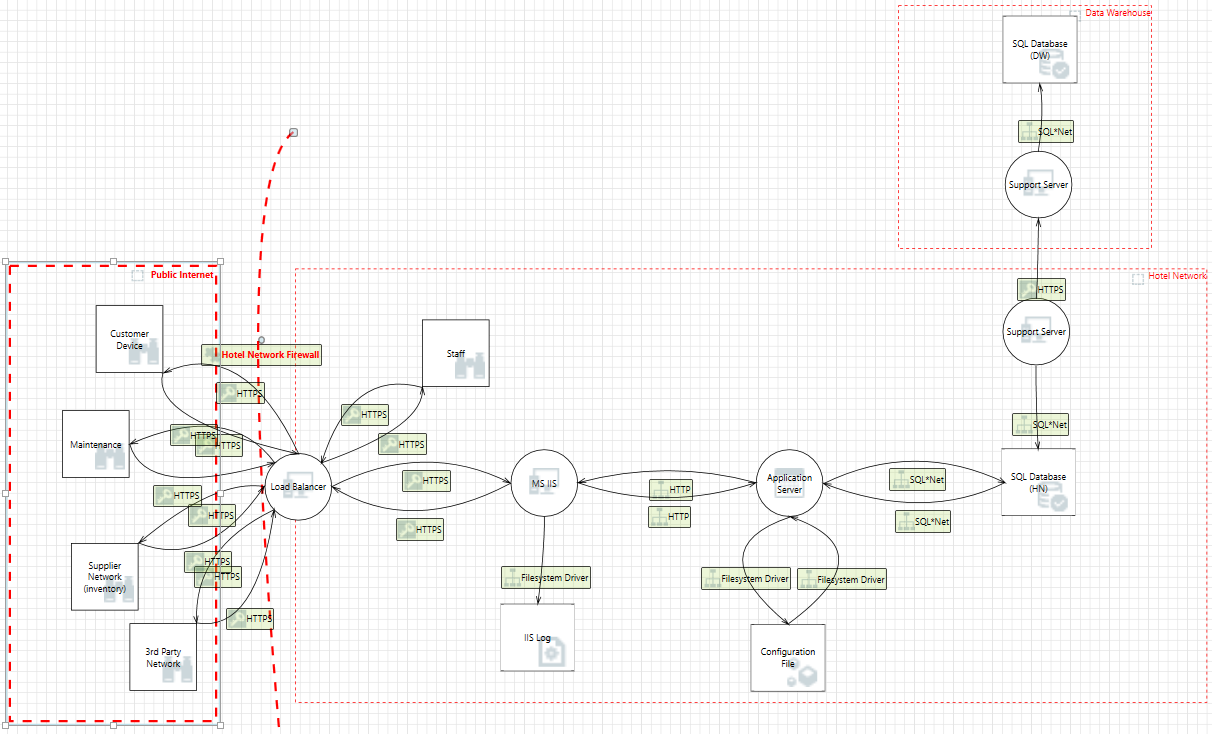
https://www.owasp.org/index.php/OWASP\_Top\_Ten\_Cheat\_Sheet

# 2. SYSTEM OVERVIEW

The web application resides behind the hotel network, which is protected by a firewall. The front end load balancer interacts with the public-facing internet, and the back end rests on servers and SQL databases contained in the hotel network and data warehouse. On the front-end, the load balancer increases availability by managing incoming client connections, blocking requests that are obvious threats early-on, and efficiently serving nonthreatening clients to the back-end using concurrency. It acts as a reverse proxy for the web application.   
 MS IIS is the server that manages the web application by serving content to the user’s browser. It will log pertinent data to meet any auditing needs. In the event of data loss or any sort of claim on behalf of users, having these logs is necessary to dispute claims and derive the truth from the situation. That being said, sufficient protections must be in place to secure this log.   
 The application server, which utilizes an obfuscated configuration file, is where the business logic takes place. It provides components and resources available for execution and creates subsequent content for the user’s browser. The application server interacts directly with a SQL database within the hotel network, which communicates with the server in the data warehouse. That server is near the data warehouse’s SQL database where the back end is stored and backed up. Most components will communicate over HTTPS while a few use SQL\*Net or local server filesystems.

# 3. SYSTEM ARCHITECTURE

## 3.1  Architectural Design



## 3.2  Decomposition Description

## ../../../Desktop/Screen%20Shot%202018-05-01%20at%202.59.53%20P

## 3.3  Design Rationale

The hotel needs to be able to communicate with public internet, so considerations for confidentiality, integrity, and availability are all necessary to include in the architectural design. Load balancer and its configuration were chosen to increase availability and prevent denial of service. Any input data from public internet is sanitized to prevent a number of control flow issues, as well as checked for anti-forgery tokens to ensure authenticity. Load balancer acts as a traffic cop to sort and manage traffic effectively in order to be the first step in risk mitigation MS IIS drives the application server and logs dataflow in order to prevent repudiation. All servers are isolated to the app container to help prevent tampering, program flow whitelisted, and data is encrypted to prevent information disclosure. Sensitive servers have been defaulted to least privilege in order to mitigate elevation of privilege, and SQL dataflow is protected by prepared statements.

## 4.1  Data Description

Load balancer processes client request data and stores it as JSON objects. It then puts them in a queue to be distributed to subsequent servers for execution. While the user is being authenticated, the server accesses that user file stored in the SQL database (or creates a new entry). After authentication, processes sent from Load balancer to MS IIS server are logged for auditing using the server’s filesystem, and then data is sent to the application server for processing. The application server consults the back-end concerning hotel room availability, user accounts, and customer reservations for each request it serves. For staff requests, slightly more flows of execution are available, as outlined above in 3.1. Room availability(1) requests retrieve data stored in the back-end, delivering the resulting content of the query to the application server, which packs it as a JSON object. It’s then sent to the MS IIS server which prepares the content for delivery. Transactions (reservations) are logged by a filesystem and stored in the SQL database and operate similarly to the aforementioned availability(1) request. Staff interactions with the system behave similarly, but with elevated privilege over the domain compared to customers’ access. Staff can see everything customers can, as well as inventory, maintenance tickets, check the state of all room reservations, and make changes to update that state.

## 4.2  Data Dictionary

|  |  |  |
| --- | --- | --- |
| **Customer** | **Data type** | **Description** |
| ID | Int | Unique identifier |
| First name | Varchar(30) | Customer’s name |
| Family name | Varchar(50) | Customer’s surname |
| Phone number | Int | Phone number to reach them |
| Credit card info | Int | Credit card #/code/exp date |
| Register date | Datetime | When did they register on site? |
| Accommodations | Varchar(500) | Disability accommodations |

|  |  |  |
| --- | --- | --- |
| **Inventory** | **Data type** | **Description** |
| Item ID | Int | Unique identifier for an item |
| Description | Varchar(100) | Describe inventory item |
| Quantity | Int | How many of said item? |

|  |  |  |
| --- | --- | --- |
| **Maintenance tickets** | **Data type** | **Description** |
| ID | Int | Unique ticket identifier |
| Department | Int | Dept ID where work is needed |
| Description | Varchar(300) | A description of problem |
| Status | Varchar(300) | Resolved/not resolved? |
| Author (employee) | Int | Who created the ticket? Emp ID |

|  |  |  |
| --- | --- | --- |
| **Reservation** | **Data type** | **Description** |
| ID | Int | Unique identifier |
| Room | Int | Room ID |
| Customer | Int | Customer ID |
| Fulfilled by | Int | Employee ID who reserved it |
| Requests/accommodations | Varchar(100) | Requests for the reservation |
| Valet service:make/model/plate | Varchar(200) | Relevant valet info |

|  |  |  |
| --- | --- | --- |
| **Room** | **Data type** | **Description** |
| ID | Int | ID represents physical location |
| Num beds | Int | How many beds are in the room |
| Bed type | Varchar(50) | What types of beds |
| Max people | Int | Max people allowed in room |
| Room type | Varchar(50) | Penthouse, large, small, etc |

|  |  |  |
| --- | --- | --- |
| **Staff** | **Data type** | **Description** |
| ID | Int | Unique identifier |
| First name | Varchar(30) | Employee name |
| Family name | Varchar(50) | Employee surname |
| Register date | Datetime | When did they join the team? |
| Email | Varchar(40) | Employee email |
| Phone number | Int | Employee phone number |
| Emergency contact # | Int | In case of emergency |
| Address | Varchar(80) | Employee address |
| Department | Int | Department ID from table Depts |
| Role | Int | Role ID from table Roles |
| Supervisor | Int | Supervisor ID from table supvs |

|  |  |  |
| --- | --- | --- |
| **Transaction** | **Data type** | **Description** |
| ID | Int | Transaction unique ID |
| Customer | Int | Customer ID |
| Reservation | Int | Reservation ID |
| Transaction date/time | Datetime | When transaction took place |
| Refundable | bit | Pay premium for refund option |

# ­­­­­5. COMPONENT DESIGN

The customer needs to make requests to the server in order to log in, to check for available hotel rooms, and to make a reservation. The staff must also be able to check for available rooms. As with any frontward-facing system, sufficient protections must be in place to mitigate potential threats from the public to ensure confidentiality, integrity, and availability. SQL injection is a high priority vulnerability to assess which involves sufficiently cleansing input of malicious code and whitelisting program execution by use of prepared statements.

*Pseudocode for protection against SQL injection*Connect to database  
Generate query such as query = “SELECT username FROM usernames WHERE username = ?”Prepare new statement(query, connection)  
Bind parameters  
Execute  
--

When staff checks a customer in, there may be some discrepancy between the storage of the reservation and the data trail of the transaction. In case of such repudiation on behalf of the SQL server (maybe it was offline at the time), all transactions are thoroughly logged and secured within the server filesystem.  
  
 *Pseudocode for secure logging*  
Receive input data as JSON object  
Receive key for encryption, store in String.  
do input.encrypt(key)  
return encrypted String data for storage   
Parse into XML  
Authenticate  
Write to hidden log  
--

In order to protect assets from attackers who try to elevate their privilege in order to impersonate staff and gain access to inventory, there would need to be protections in place concerning methods of attaining elevated privilege. If an attacker is someone who could get physical access to supply cabinets (or knows someone who could get physical access), they would be able to conveniently sift through a list of items they want to take or don’t want to take. Buffer overflow is a common method of gaining access to data that the public doesn’t normally have access to. In order to prevent these attacks we should avoid using functions that expose details of the buffer, avoid disclosing sensitive information in error messages, and range check input values to make sure they fit with the intended execution flow.

*Pseudocode for range checking  
 line by line example*  
char buffer[1024];  
fgets(buffer, sizeof buffer, stdin);

or alternatively:

*Pseudocode- character by character example*  
BUFFSIZE = 1024;  
buffer[BUFFSIZE] = {0};  
int ch;  
int i = 0;  
while ((ch = getchar()) != /n or EOF && i < BUFFSIZE -1)  
 buffer[i++] = ch;  
--

An attacker who wants to cause costly trouble for the hotel could do so by muddying the maintenance ticket user flow. Weak password policy along with an insufficiently secured password files could result in such a disruption. In order to maintain the authenticity and integrity of the data associated with maintenance tickets, password data, and other internal hotel data, staff should make sure their passwords are a complete pass phrase that contains at least one capital letter, a number, and a special character. Every 3 months, staff will be required to change their pass phrase. Two factor authentication may be useful to make sure only specialized & trusted staff read and write the maintenance tickets.

*Pseudocode for two factor authentication*Request username and password  
Process first factor  
Request alphanumerical string sent to user-owned device (or a device owned by the business designated to only that person)  
Process string  
Authenticate  
if(Authentication fails)  
 Mark down strikes i of 4.   
 if(strikes = 4)  
 lock login page  
 refuse connection  
 wait for admin credentials  
--

In the case that an attacker spoofs the server which the customer is connected to, the full contents of their transaction may be made available to the attacker. In order to protect against spoofing attacks, we can implement a strong authentication like using signed certificates.

*Pseudocode*if (hasCertificate)  
 evaluate who issued the certificate  
 Evaluate who the certificate is issued to  
 Evaluate who the certificate is issued to  
 Evaluate the expiration date

If(all tests pass)  
 continue execution flow  
else  
 close connection  
--

# 6. HUMAN INTERFACE DESIGN

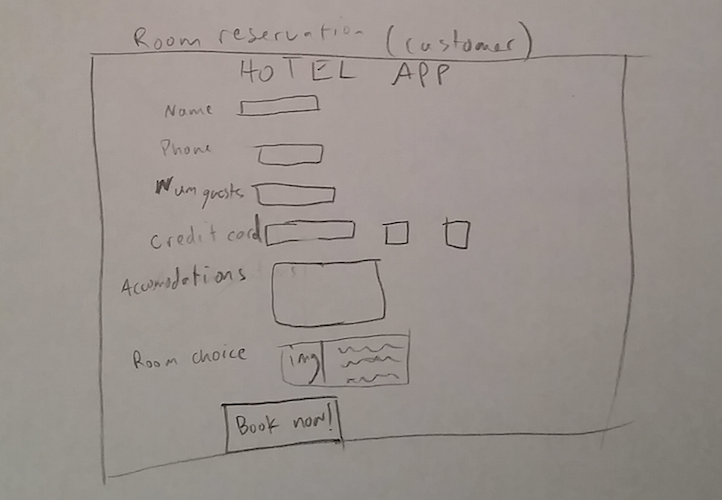
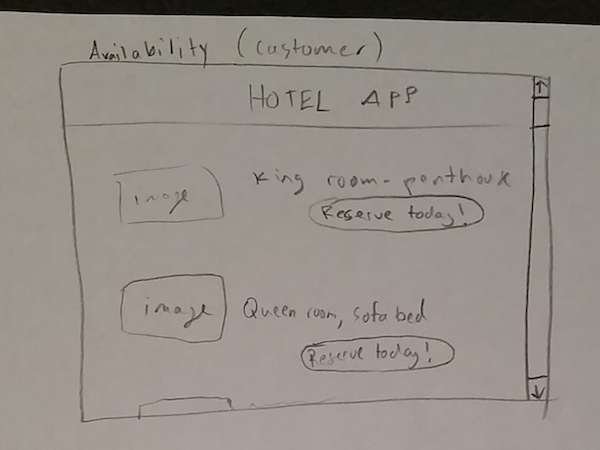
## 6.1  Overview of User Interface

Customers will have the ability to log in and search for available rooms within the hotel. Based on the availability, they will be able to make reservations. The user will be presented with a splash page from which they log in to provide authentication in order to enter the site. Then they will have access to certain resources pertaining to their privilege level.

Staff will be able to see more in addition to room availability: they will see a maintenance ticket & tracking module, customer check in/check out module, as well as inventory checking and modification.

## 6.2  Screen Objects and Actions

## ../../../Desktop/login.png



# ../../../Desktop/staff_splash.png../../../Desktop/Availability(staff).jpg../../../Downloads/20180502_215459%20(1).jpg../../../Desktop/checkinout(staff).png../../../Desktop/inventory.png../../../Desktop/maint.png