

**Experiment 1: Write the problem statement, its purpose, scope and literature review. The problem statement is intended for a broad audience and should be written in non-technical terms.**

## **1. Introduction**

### **1.1 Purpose**

The purpose of creating the system is improving the customer service by providing convenience to customer to make booking through system and increase customers. Besides it enables customers to get every bit of information regarding facilities. By this way it can save transport, calling charges and time. The main objective of the entire activity is to automate the process of day-to-day activities of hotel like:

1. Room activities
2. Admission of a New Customer
3. Assign a room according to customer's demand
4. Checkout of a computer and releasing the room
5. Compute the bill
6. Packages available
7. Advance online bookings
8. Online Cancellation
9. Email facility
10. Feedbacks.

### **1.2 Intended Audience and Reading Suggestions**

This document is to be read by the development team, the project managers, marketing staff, testers, and documentation writers. The software engineer/Developer and project managers need to become intimately familiar with the SRS. Others involved need to review the document.

Testers need an understanding of the system features to develop meaningful test cases and give useful feedback to the developers. The developers need to know the requirements of the software product they need to build.

This document is for general discussions on the implementation decisions regarding the Hotel Reservation System. The user of the product should have the concepts of RDMS, SQL, interfaces, and classes.

### **1.3 Product Scope**

This project will consist of creating an application mainly for reserving a hotel room. Modules of the software will include booking and viewing rooms & amenities for users to give them flexibility and portability. By just a few clicks users can book room and amenities of the Hotel.

## **2. Overall Description**

### **2.1 Product Perspective**

In this project, we have tried to solve the problem of booking a hotel room by computerizing the whole system of booking a room. We have tried to make a web-based application in which a user can login and see the available room and book the room or cancel the room booked by the user. Then the confirmation is received which is done by the manager of the hotel from admin side. In this way the whole system of booking is computerized.

### **2.2 Product Functions**

The general functions of our product are:

1. View available rooms
2. Book available rooms
3. Cancel booked rooms

### **2.3 User Classes and Characteristics**

There are two user levels in our system

- I. Manager
- II. Customer/User

#### **Manager:**

Manager is responsible for conformation of the booking of the customer. Manager doesn't have privileges to access any user's personal data. Manager can see booking details of the customer and can send an email to confirm the booking.

#### **User:**

User can access the main functionalities of the system. User will be able to book rooms as well as amenities and can pay for it from the app itself.

### **2.4 Operating Environment**

The hotel reservation system is a work on securing online booking through the hotel's website. Then, the data will be sent to a backend system that can reach the hotel for reservation. For example, you can provide other features, such as automation of reserved emails.

### **2.5 Design and Implementation Constraints**

Best effort is provided by the software development team in developing the software. In order to maintain the reliability and durability of system, some design and implementation constraints are applied. It can only accommodate 1000 concurrent users. Booking can also be done via offline mode, so any failure in offline system can affect the integrity of the website.

**Experiment 2: Develop requirements specification for a above problem (The requirements specification should include both functional and non-functional requirements) and also includes the constraints.**

## **1. External Interface Requirements**

### **User Interfaces**

1. JavaScript enabled Web Browser
2. Internet Connection

### **Hardware Interfaces**

1. CPU: 1.4 GHz Dual Core processor
2. RAM: 2GB
3. Platforms: Windows, Linux or Mac

### **Software Interfaces**

1. PHP
2. JavaScript
3. Lucidchart.com

### **Communications Interfaces**

1. HTTP/HTTPS
2. TCP/IP

## **2. System Features**

Our system contains following key features:

### **1. View room:**

Users / Customers, after they login into the system, are able to access the functionality of our web application through their browser. Users can view the available rooms of the hotel, also they can search for the hotels as per their requirement. Users can select the number of people they want to reserve the room for and for how many days. After the user have selected the options which meet their requirement, they can proceed for room reservation which is then verified by the employee (manager) of the hotel.

### **2. Book Rooms:**

The rooms which are available for booking can be booked by the user with a single click. But in case, if the room is already taken or not in good condition or due any other hotel room inconvenience, the user will not be able to book that particular room. The acceptance or rejection of the room selected by the user will be determined by the employee (manager) of the hotel.

### 3. **Cancel booking:**

User can cancel the room booked earlier by logging in to the web application and selecting the “cancel booked room” option. It is also possible for the user to cancel the booked room by calling to the hotel but it should be avoided as much as possible as it is not the standard method of doing so. The user may select a reason for canceling the booked room.

### 4. **Confirm status:**

The employee of hotel (manager) completes the final process of hotel room reservation either by confirming the hotel room checked by the user or canceling the hotel room. In any case, the action taken by the hotel employee will be final. Usually, the employee will accept the room reservation. A message will be sent to the user after the employee has confirmed the room status.

### 5. **Billing:**

Once the hotel room reservation is complete, the user will get the total bill amount messaged into their account through web application. The user can pay after they arrive at the hotel.

### 6. **Report Generation:**

A report is generated at the admin side which includes total number of room bookings, number of customers, number of room cancelation, number of available rooms, number of repeated customers, customer feedback, web application traffic, and so on.

## **Functional Requirements of the Hotel Reservation System**

The functional requirements of this system are:

- Register new customer
- Book rooms
- Cancel booked rooms
- Confirm status

## **Non-Functional Requirements of the Hotel Reservation System**

In this system, the authentication of the manager is an important factor. In this system, manager authentication will be done by login by user name and password and classified by user type. The system has a consistent interface so that the system is easy to use.

### **3 Other Nonfunctional Requirements**

#### **a. Performance Requirements**

1. Login Validation should be done immediately.
2. Response to customer inquiry must be done within ten minutes.

#### **b. Safety Requirements**

Under failure, system should be able to come back at normal operation under an hour.

#### **c. Security Requirements**

All external communications between the data's server and client must be encrypted.

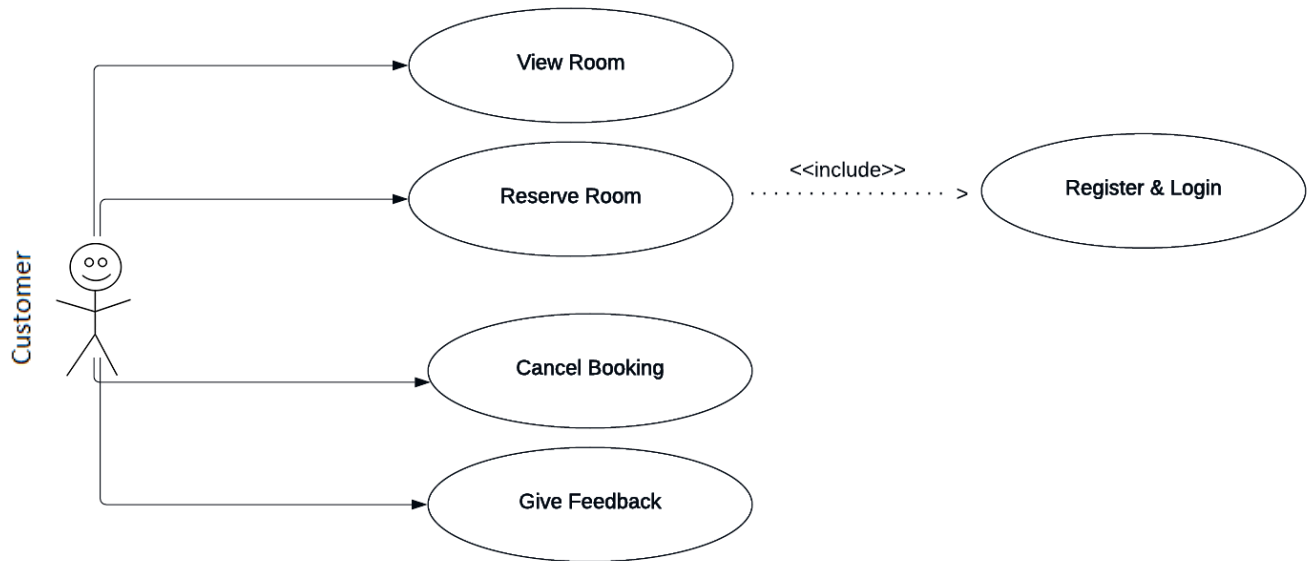
#### **d. Software Quality Attributes**

- Flexibility: System should be built flexible enough to add new features and integrate with external system API's from booking.com or expedia.com in future.
- Integrity: System should secure customer's details to avoid data losses and data manipulation
- Usability: System should be able to easily accessible by all users
- Maintainability: System shall be able to maintain easily at any point of time in future
- Testability: System shall be able to test and confirm all the specifications according to client's requirements.

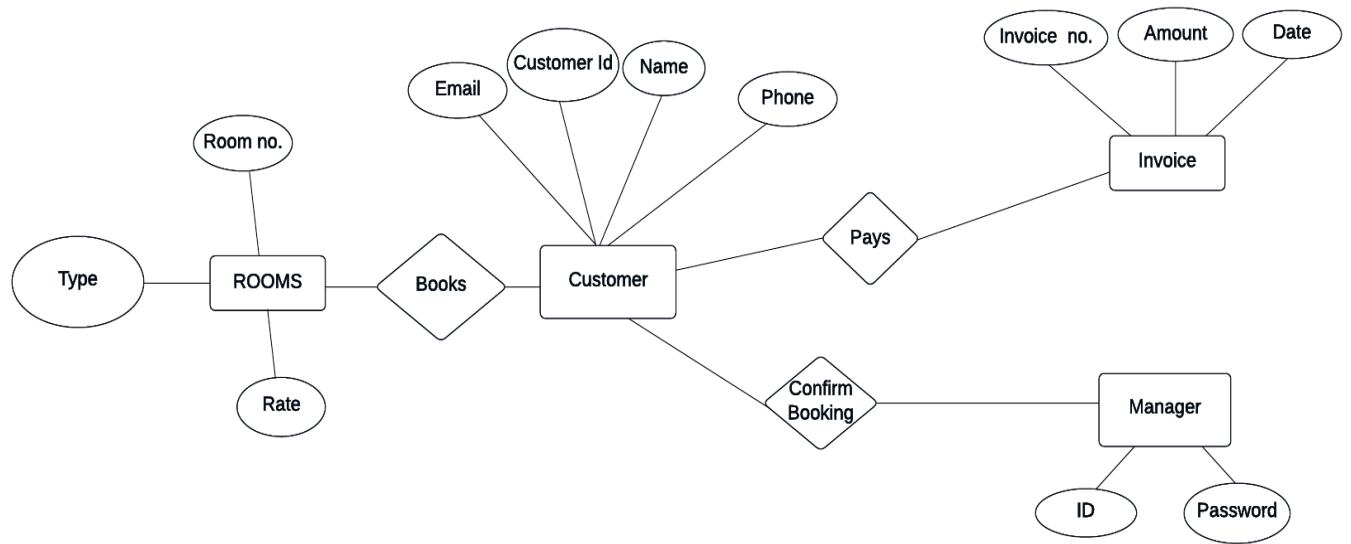
#### **e. Business Rules**

- Room Type: Single Room; Double Room.  
1 single room can accommodate either 1 adult or 1 child. 1 double room can accommodate 2 adults, 2 children or 1 adult and 1 child.
- GDPR implies deleting Customer's Credit card information within 24 hrs. after checkout.

**Experiment 3: To perform the user's view analysis for the suggested system - Draw the use case diagram.**



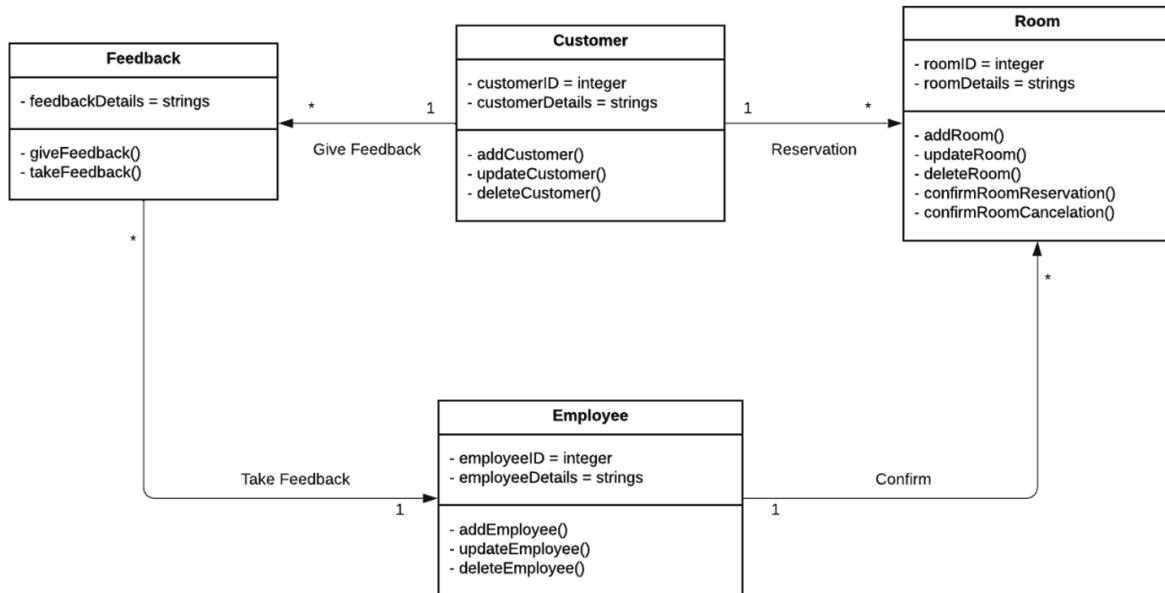
**Experiment 4: Design function-oriented diagram for the selected system - Draw the E-R Diagrams.**



## Experiment 5: Design structural view diagram for the selected system.

### i) Class Diagram

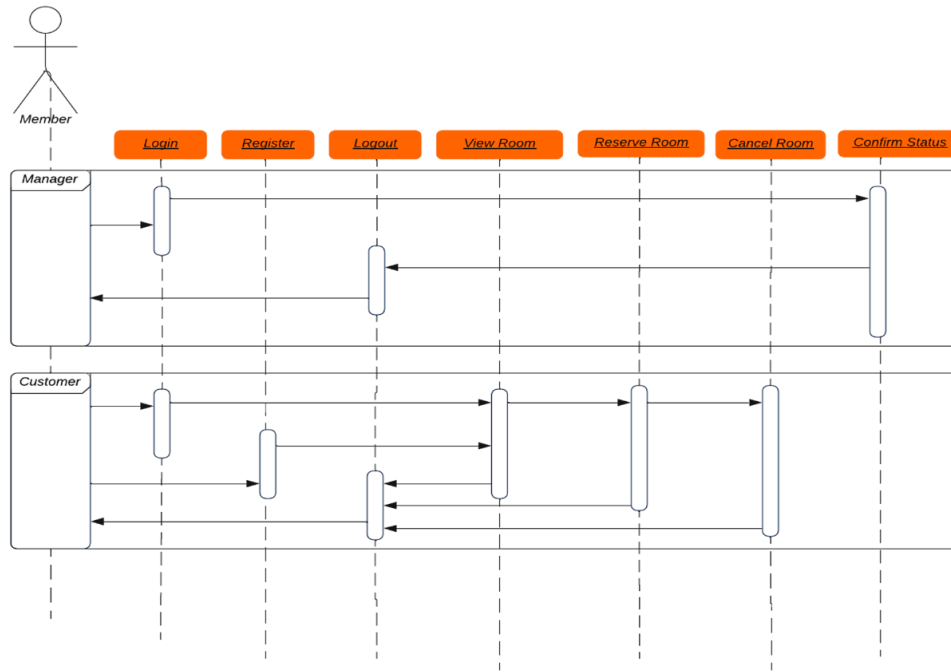
### ii) Object Diagram



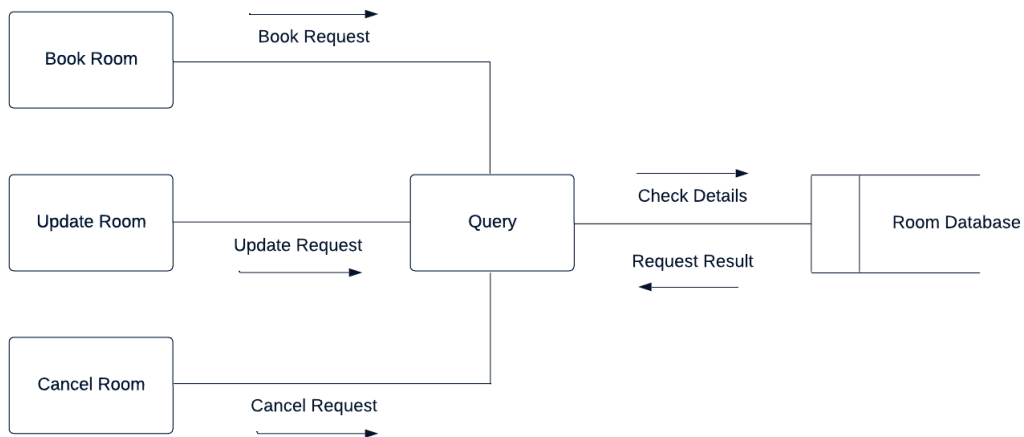


## Experiment 6: Design the behavioral view diagram for the selected system - Interaction Diagrams

### i) Sequence Diagrams

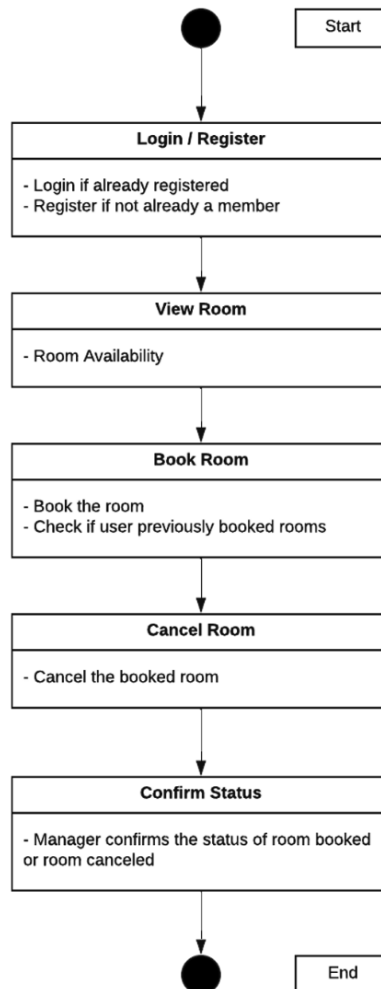


### ii) Collaboration Diagram

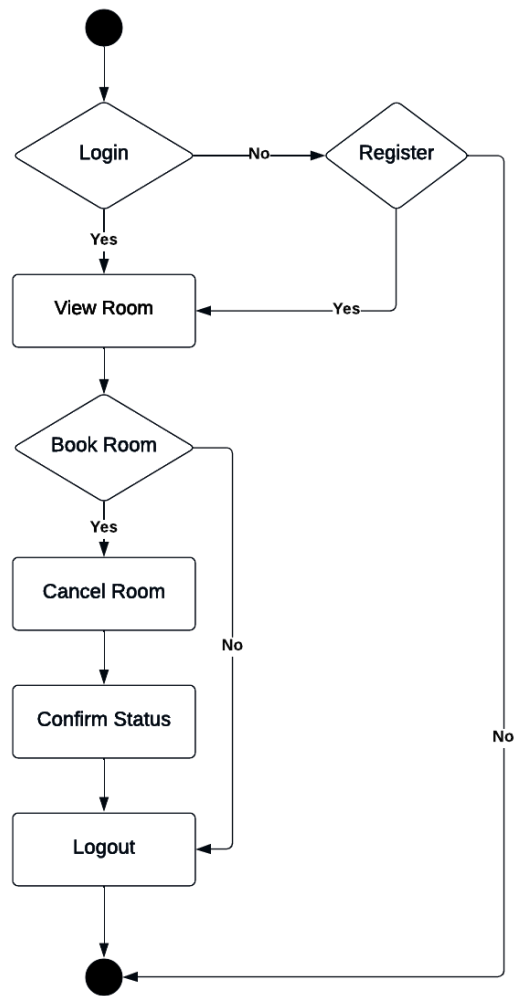


## Experiment 7: Design the behavioral view diagram for the selected system.

### i) State-Chart Diagrams

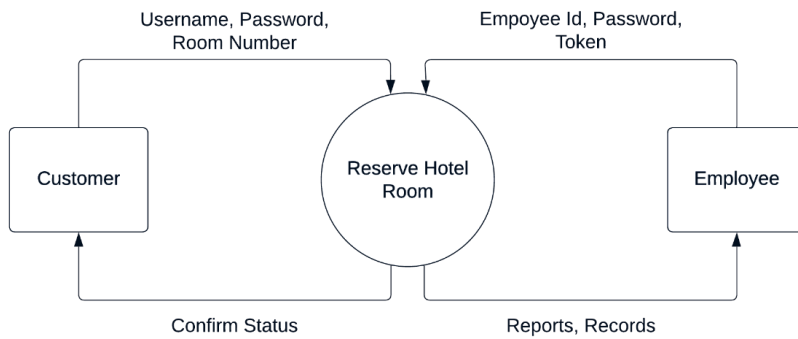


## ii) Activity Diagrams

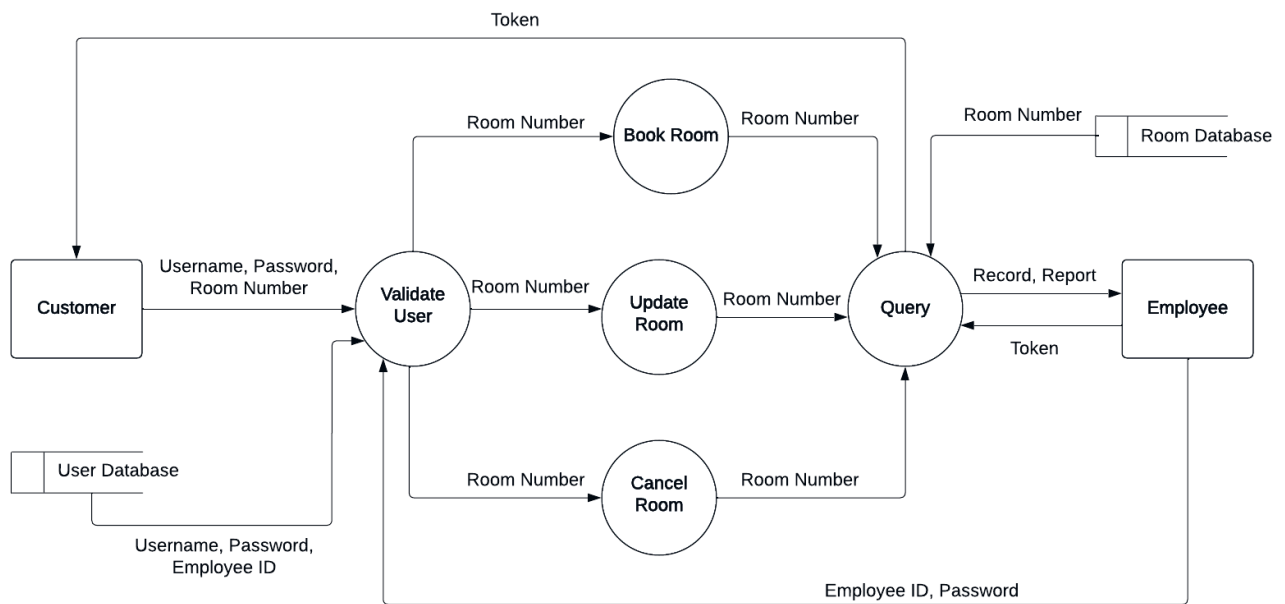


## Experiment 8: Develop Structured design for the DFD model for the selected system.

### Level 0 DFD:



### Level 1 DFD:



## **Experiment 9: Implementation Planning and details for Selected system.**

### **i) Implementation Environment (Single vs Multiuser, GUI vs non-GUI)**

It is a web-based multi-user hotel reservation solution for a hotel.

### **ii) Program/Modules Specification**

- User Authentication
- Room Management
- Integrated Manager Verification

### **iii) Security Features**

Access to the various subsystems is protected by a user login screen that requires an email address and password. Your data is stored in Google Firebase with a high level of protection. To protect user privacy, passwords are encrypted with a hash key.

### **iv) Sample Coding and Screen shots**

We have used some of the following coding standards:

- Proper indentation is used
- Complex coding style is avoided
- Length of functions are limited to an extent
- Avoiding GOTO statements
- Naming conventions for local variables, global variables, constants and functions are strictly followed

## Experiment 10: Do Testing (choose appropriate testing strategy or techniques suitable to your system)

### i) Testing Plan

- Cross-Browser Compatibility Testing
- Application's Performance Under Various Conditions
- Validate all Security Issues with Security Testing

### ii) Testing Methods

- Page validations with and without JavaScript enabled
- All images and alignment
- Page layout in different resolutions
- Dedicated test environment for load testing
- Predict the number of users that will access the system
- Open Web Application Security Project (OWASP)
- Penetration Testing Execution Standard (PTES)

### iii) Test Cases (Purpose, required output, Expected Result)

The purpose of this testing is to verify the functionality and usability of web application.

#### User Authentication:

Test Case	Expected Result
Login Button	User must be able to login to their profile by entering id and password
Register Button	It opens form for new user for registration
Submit Registration	It submits the user credential to the database and created user profile

#### Room Management:

Test Case	Expected Result
Search Button	User must be able to search available rooms
Book Button	User must be redirected to validate hotel room reservation
Confirm Booking	User must be able to confirm the chosen room
Cancel Booking	User must be able to cancel the booked room

#### Integrated Manager Verification:

Test Case	Expected Result
Accept Booking	Manager should be able to accept the booking request
Reject Booking	Manager should be able to reject the booking request
Accept Cancellation	Manager should be able to accept the cancellation request
Reject Cancellation	Manager should be able to reject the cancellation request

## **Experiment 11: Specify the Limitation, Future Enhancement and Conclusion for selected system.**

### **Limitations:**

- The system cannot handle more than 1,000 concurrent users at a time.
- The web application will not perform as intended if JavaScript is blocked on the client browser.

### **Future Scope:**

- The primary ToDo plan is to increase the number of users the system can support
- To decrease the web application dependency on JavaScript for important functionality
- To integrate payment system within the web application

### **Conclusion:**

This project aims to create a hotel reservation system that customers can use to book hotel rooms. Users can view rooms, can register for membership and log in. Managers can see the user's daily reservation history. The purpose of creating these applications is to speed up people's work and save users' time. The aim is to create healthy user experience and increase customer for hotels.

## Experiment 12: Introduction to GIT and account creation on GIT

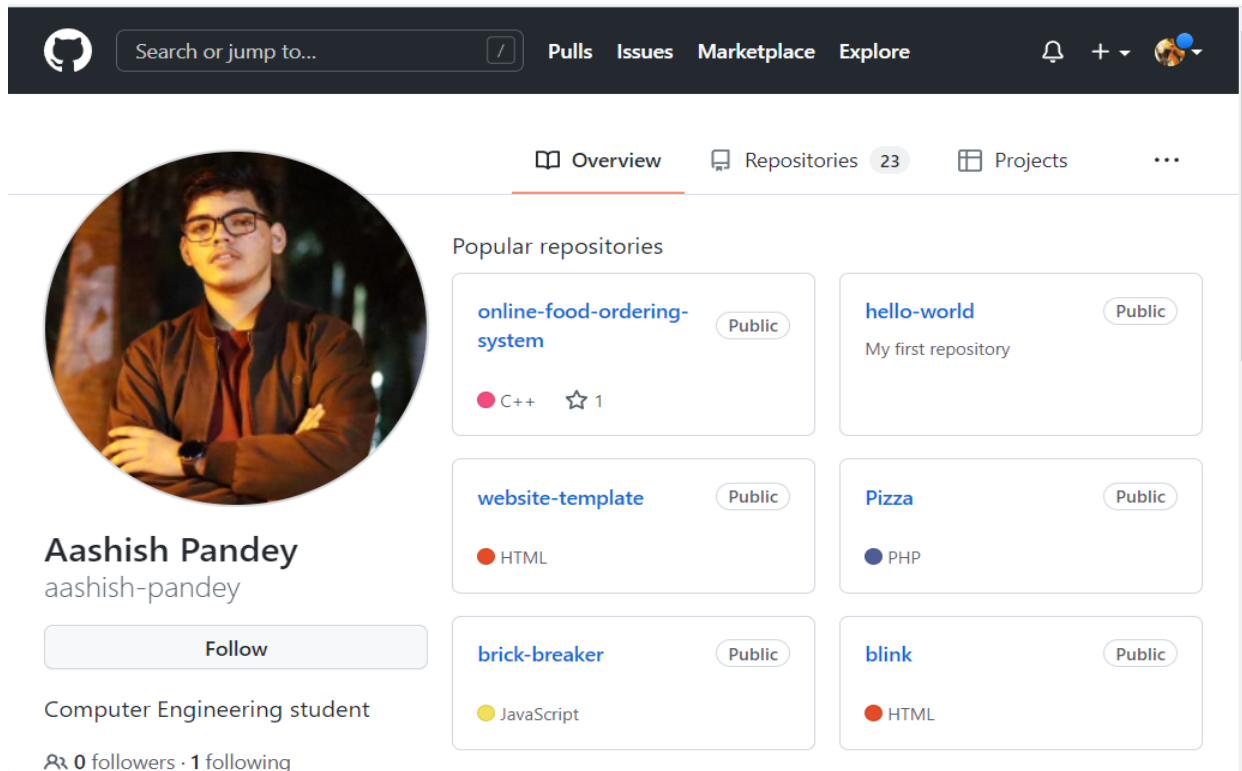
Git is a popular version control system. It was created by Linus Torvalds in 2005, and has been maintained by Junio Hamano since then.

It is used for:

- Tracking code changes
- Tracking who made changes
- Coding collaboration

What does Git do?

- Manage projects with Repositories
- Clone a project to work on a local copy
- Control and track changes with Staging and Committing
- Branch and Merge to allow for work on different parts and versions of a project
- Pull the latest version of the project to a local copy
- Push local updates to the main project



The screenshot shows the GitHub profile of Aashish Pandey. The profile includes a circular profile picture of a man with glasses and a brown jacket. Below the picture, the name 'Aashish Pandey' is displayed, followed by the username 'aashish-pandey' and a 'Follow' button. The bio 'Computer Engineering student' is also visible. The profile shows 0 followers and 1 following. The 'Overview' tab is selected, displaying a list of popular repositories. The repositories listed are: 'online-food-ordering-system' (C++, 1 star), 'hello-world' (My first repository), 'website-template' (HTML), 'Pizza' (PHP), 'brick-breaker' (JavaScript), and 'blink' (HTML). Each repository card indicates it is 'Public'.

GitHub Link: <https://github.com/aashish-pandey>



## **Experiment 13: Introduction to Team Foundation server tool**

Team Foundation Server (often abbreviated as TFS) is a Microsoft product that provides tools and technologies designed to help teams collaborate and coordinate their efforts to complete a project or create a product.

This includes DevOps capabilities across the entire application lifecycle. Core elements include source code management, requirements management, project management, reporting, test and release management capabilities. TFS is tailored for Visual Studio and Eclipse, but is also used as a backend in other IDEs. Let's take a closer look at this powerful product. The main function of Team Foundation is to make it easier for teams to work together to complete a product or project. When working in TFS, the central theme is team projects.

Each project is stored on Team Foundation Server, providing a central location for teams to access and coordinate their work. To facilitate this collaboration, each team project shares a Windows SharePoint Web site, the Project Portal. It allows teams to collaboratively store and share documents, control document versions, and use collaborative features such as lists and calendars.

## Experiment 14: Study of Testing Tool: Load Runner

LoadRunner is a Performance Testing tool which was pioneered by Mercury in 1999. LoadRunner was later acquired by HPE in 2006. In 2016, LoadRunner was acquired by Microfocus. LoadRunner supports various development tools, technologies and communication protocols. In fact, this is the only tool in market which supports such a large number of protocols to conduct Performance Testing. Performance Test Results produced by LoadRunner software are used as a benchmark against other tools. LoadRunner is not only pioneer tool in Performance Testing, but it is still a market leader in the Performance Testing paradigm. In a recent assessment, LoadRunner has about 85% market share in Performance Testing industry.



Broadly, LoadRunner tool supports RIA (Rich Internet Applications), Web 2.0 (HTTP/HTML, Ajax, Flex and Silverlight etc.), Mobile, SAP, Oracle, MS SQL Server, Citrix, RTE, Mail and above all, Windows Socket. There is no competitor tool in the market which could offer such wide variety of protocols vested in a single tool.

What is more convincing to pick LoadRunner in software testing is the credibility of this tool. LoadRunner tool has long established a reputation as often you will find clients cross verifying your performance benchmarks using LoadRunner. You'll find relief if you're already using LoadRunner for your performance testing needs. LoadRunner software is tightly integrated with other HP Tools like Unified Functional Test (QTP) & ALM (Application Lifecycle Management) with empowers you to perform your end-to-end Testing Processes. LoadRunner works on a principal of simulating Virtual Users on the subject application. These Virtual Users also termed as VUsers, replicate client's requests and expect a corresponding response to passing a transaction. Why do you need Performance Testing? An estimated loss of 4.4 billion in revenue is recorded annually due to poor web performance. In today's age of Web 2.0, users click away if a website doesn't respond within 8 seconds. Imagine yourself waiting for 5 seconds when searching for Google or making a friend request on Facebook. The repercussions of performance downtime are often more devastating than ever imagined. We've examples such as those that recently hit Bank of America Online Banking, Amazon Web Services, Intuit or Blackberry. According to Dunn & Bradstreet, 59% of Fortune 500 companies experience an estimated 1.6 hours of downtime every week. Considering the average Fortune 500 company with a minimum of 10,000 employees is paying \$56 per hour, the labor part of downtime costs for such an organization would be \$896,000 weekly, translating into more than \$46 million per year. Only a 5-minute downtime of Google.com (19-Aug-13) is estimated to cost the search giant as much as \$545,000. It's estimate that companies lost sales worth \$1100 per second due to a recent Amazon Web Service Outage.

When a software system is deployed by an organization, it may encounter many scenarios that possibly result in performance latency. A number of factors cause decelerating performance, few examples may include:

- Increased number of records present in the database
- Increased number of simultaneous requests made to the system
- A larger number of users accessing the system at a time as compared to the past

**LoadRunner Architecture Diagram** Suppose you are assigned to check the performance of Amazon.com for 5000 users in a real-life situation, these all these 5000 users will not be at homepage but in a different section of the websites. How can we simulate differently? VUGen: VUGen or Virtual User Generator is an IDE (Integrated Development Environment) or a rich coding editor. VUGen is used to replicate System Under Load (SUL) behavior. VUGen provides a “recording” feature which records communication to and from client and Server in form of a coded script – also called VUser script. So, considering the above example, VUGen can record to simulate following business processes:

1. Surfing the Products Page of Amazon.com
2. Checkout
3. Payment Processing
4. Checking MyAccount Page

**Controller** Once a VUser script is finalized, Controller is one of the main LoadRunner components which controls the Load simulation by managing, for example:

- How many VUsers to simulate against each business process or VUser Group
- Behavior of VUsers (ramp up, ramp down, simultaneous or concurrent nature etc.)
- Nature of Load scenario e.g., Real Life or Goal Oriented or verifying SLA
- Which injectors to use, how many VUsers against each injector
- IP Spoofing
- Error reporting
- Transaction reporting etc.

Taking an analogy from our example controller will add the following parameter to the VUGen Script

- 1) 3500 Users are Surfing the Products Page of Amazon.com
- 2) 750 Users are in Checkout
- 3) 500 Users are performing Payment Processing
- 4) 250 Users are Checking MyAccount Page ONLY after 500 users have done Payment Processing

Even more complex scenarios are possible:

1. Initiate 5 VUsers every 2 seconds till a load of 3500 VUsers is achieved.
2. Iterate for 30 minutes
3. Suspend iteration for 25 VUsers
4. Re-start 20 VUsers
5. Initiate 2 users (in Checkout, Payment Processing, MyAccounts Page) every second.
6. 2500 VUsers will be generated at Machine A
7. 2500 VUsers will be generated at Machine B