A Laboratory Manual for

SOFTWARE ENGINEERING (3150711)

B.E. Semester V

(Department of Computer Engineering)

Directorate of Technical Education, Gandhinagar, Gujarat.

Government Engineering College, Modasa

Certificate

This is to certify that Mr./Ms. ____ ________ Enrollment No. ________________ of B.E. Semester _____ Computer Engineering of this Institute (GTU Code: 016) has satisfactorily completed the Practical / Tutorial work for the subject **Software Engineering (3150711)** for the academic year 2022-23.

Place: ___________________

Date: ___________________

Name and Sign of Faculty member

Head of the Department

Preface

The main motto of any laboratory/practical/field work is to enhance required skills and create ability amongst students to solve real-time problems by developing relevant competencies in the psychomotor domain. By keeping this in view, GTU has designed a competency-focused outcome-based curriculum for engineering degree programs where sufficient weightage is given to practical work. It shows the importance of enhancement of skills amongst the students, and it pays attention to utilizing every second of time allotted for practicals amongst students, instructors, and faculty members to achieve relevant outcomes by performing the experiments rather than merely study-type experiments. It is a must for the effective implementation of a competency-focused outcome-based curriculum that every practical is keenly designed to serve as a tool to develop and enhance relevant competency required by the various industry among every student. These psychomotor skills are very difficult to develop through traditional chalk-and-board content delivery methods in the classroom. Accordingly, this lab manual is designed to focus on industry-defined relevant outcomes rather than the old practice of conducting practicals to prove concepts and theories.

By using this lab manual, students can go through the relevant theory and procedure in advance before the actual performance, which creates interest, and students can have a basic idea prior to the performance. This, in turn, enhances pre-determined outcomes amongst students. Each experiment in this manual begins with competency, industry-relevant skills, course outcomes as well as practical outcomes (objectives). The students will also achieve safety and necessary precautions to be taken while performing practicals.

This manual also provides guidelines to faculty members to facilitate student-centric lab activities through each experiment by arranging and managing necessary resources in order that the students follow the procedures with required safety and necessary precautions to achieve the outcomes. It also gives an idea of how students will be assessed by providing rubrics.

Software Engineering is an application of a systematic, defined, and measurable approach that begins with requirement specification and progresses with planning, modeling, and testing, and concludes with deployment. It is a layered paradigm that comprises processes, methods, and tools with the bedrock of quality focus. The Software Engineering approach's main purpose is committed to developing the software products within the stipulated time and budget with more quality. Quality product motivates firmness, commodity, and delight.

Utmost care has been taken while preparing this lab manual; however, there is always a chance of improvement. Therefore, we welcome constructive suggestions for improvement and removal of errors, if any.

Practical – Course Outcome matrix

Course Outcomes (COs):

CO-1: Prepare SRS (Software Requirement Specification) document and SPMP (Software Project Management Plan) document.

CO-2: Apply the concept of Functional Oriented and Object-Oriented Approaches for Software Design.

CO-3. Recognize how to ensure the quality of software products, different quality standards, and software review techniques.

CO-4. Apply various testing techniques and test plans in.

CO-5. Able to understand modern Agile Development

Sr. No.	Objective(s) of Experiment	CO 1	CO2	CO3	CO4	CO5
1.	Define the Problem statement of Software development, and after identifying the requirements based on the requirement engineering tasks, prepare the Software Requirement Specification (SRS) document.	√				√
2.	Prepare the Software Project Management Plan (SPMP), including the following: • Estimation of Size, Cost, Duration, Effort • Prepare the Schedule, Milestones using the Gantt chart	√				√
3.	Prepare the following components of the Data Flow Model: • Data Dictionary • Data Flow Diagram Structure Chart		√			√
4.	Prepare the user's view analysis: Describe different scenarios and Draw Use case diagrams using UML		√			√
5.	Prepare the structural view: Draw the Class diagram and object diagram.		√			√
6.	Prepare the behavioral view: Draw a Sequence diagram and a Collaboration diagram.		√			√
7.	Prepare the behavioral view: Draw a State-chart diagram and Activity diagram.		√			√
8.	Prepare the implementation view: Draw the Component and Deployment diagram.		√			√
9.	Prepare the quality management plan.			√		√
10.	To perform various testing using the testing tool unit testing, integration testing design test cases.				√	√
11	Case Study: Study of Open-source tools in DevOps for Infrastructure Automation, Configuration Management, Deployment Automation, Performance Management, and Log Management. Monitoring					√

Index
(Progressive Assessment Sheet)

Sr. No.	Objective(s) of Experiment	Page No.	Date of performance	Date of submission	Assessment Marks	Sign. of Teacher with date	Remarks
1	Define the Problem statement of Software development and after identifying the requirements based on the requirement engineering tasks prepare the Software Requirement Specification (SRS) document.						
2	Prepare the Software Project Management Plan (SPMP) including following: • Estimation of Size, Cost, Duration, Effort • Prepare the Schedule, Milestones using Gantt chart						
3	Prepare the following components of Data Flow Model: • Data Dictionary • Data Flow Diagram Structure Chart						
4	Prepare the user's view analysis: Describe different scenarios and Draw Use case diagrams using UML						
5	Prepare the structural view: Draw Class diagram and object diagram using UML.						
6	Prepare the behavioral view: Draw Sequence diagram and Collaboration diagram using UML.						
7	Prepare the behavioral view: Draw State-chart diagram and Activity diagram using UML.						
8	Prepare the implementation view: Draw Component and Deployment diagram using UML.						
9.	Prepare the quality management plan.						
10.	To perform various testing using the testing tool unit testing, integration testing design test cases.						
11.	Case Study: Study of Open-source tools in DevOps for Infrastructure Automation, Configuration Management, Deployment Automation, Performance Management, Log Management. Monitoring						
Total							

GUJARAT TECHNOLOGICAL UNIVERSITY, AHMEDABAD,

COURSE CURRICULUM

COURSE TITLE: Software Engineering

(Code: 3150711)

Degree Program in which this course is offered	**Semester in which offered**
Computer Engineering	5th Semester

1. RATIONALE

- To study Software Development Life Cycle, Development models and Agile Software development.
- To study fundamental concepts in software testing, including software testing objectives, process, criteria, strategies, and methods.
- To discuss various software testing issues and solutions in software unit test; integration, regression, and system testing.
- To learn the process of improving the quality of software work products.
- To gain the techniques and skills on how to use modern software testing tools to support software testing projects.
- To expose Software Process Improvement and Reengineering

2. COMPETENCY

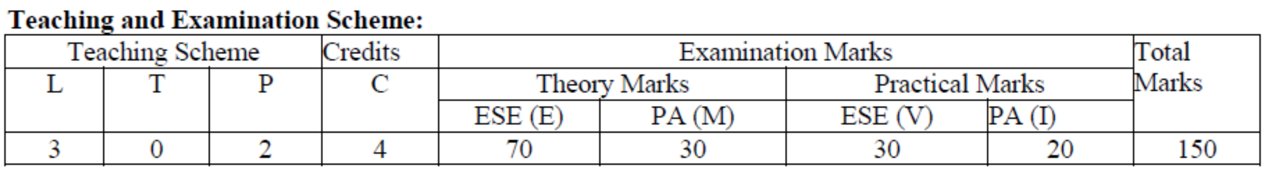
- The course content should be taught and analyze with the aim to develop different types of skills so that students are able to acquire following competency:
- Software engineering models to development of the complex engineering software based on the software development life cycle.

3. COURSE OUTCOMES

After learning the course, the students should be able to:

1. Prepare SRS (Software Requirement Specification) document and SPMP (Software Project Management Plan) document.
2. Apply the concept of Functional Oriented and Object-Oriented Approach for Software Design.
3. Recognize how to ensure the quality of software product, different quality standards and software review techniques.
4. Apply various testing techniques and test plan in.
5. Able to understand modern Agile Development.

4. TEACHING AND EXAMINATION SCHEME



Teaching and Examination Scheme:

Teaching Scheme			Credits	Examination Marks				Total Marks
				Theory Marks		Practical Marks		
L	T	P	C	ESE (E)	PA (M)	ESE (V)	PA (I)	
3	0	2	4	70	30	30	20	150

5. SUGGESTED LEARNING RESOURCES

A. LIST OF BOOKS

1. Roger S.Pressman, Software Engineering- A practitioner's Approach, McGraw-Hill International Editions
2. Ian Sommerville, Software engineering, Pearson education Asia
3. Pankaj Jalote, Software Engineering – A Precise Approach Wiley
4. Behhforoz & Frederick Hudson, Software Engineering Fundamentals, OXFORD
5. Rajib Mall, Fundamentals of software Engineering, Prentice Hall of India.
6. Deepak Gaikwad, Viral Thakkar, DevOps Tools from Practitioner's ViewPoint, Wiley
7. Merlin Dorfman (Editor), Richard H. Thayer (Editor), Software Engineering
8. Robert C. "Uncle Bob" Martin, Clean Architecture: A Craftsman's Guide to Software Structure and Design

B. LIST OF SOFTWARE / LEARNING WEBSITES

- www.onesmartclick.com/engsineering/software-engineering.html
- www.sei.cmu.edus
- https://www.edx.org/school/uc-berkeleyx
- https://devops.com/most-popular-open-source-devops-tools/
- https://www.guru99.com/devops-tutorial.html

Industry Relevant Skills

The following industry relevant competency are expected to be developed in the student by undertaking the practical work of this laboratory.

1. Apply knowledge of Machine Learning to solve real world problems
2. Understand and analyze the data analytically to use them wisely to build the ML model

Guidelines for Faculty members

1. Teacher should provide the guideline with demonstration of practical to the students with all features.
2. Teacher shall explain basic concepts/theory related to the experiment to the students before starting of each practical
3. Involve all the students in performance of each experiment.
4. Teacher is expected to share the skills and competencies to be developed in the students and ensure that the respective skills and competencies are developed in the students after the completion of the experimentation.
5. Teachers should give opportunity to students for hands-on experience after the demonstration.
6. Teacher may provide additional knowledge and skills to the students even though not covered in the manual but are expected from the students by concerned industry.
7. Give practical assignment and assess the performance of students based on task assigned to check whether it is as per the instructions or not.
8. Teacher is expected to refer complete curriculum of the course and follow the guidelines for implementation.

Instructions for Students

1. Students are expected to carefully listen to all the theory classes delivered by the faculty members and understand the COs, content of the course, teaching and examination scheme, skill set to be developed etc.
2. Students shall organize the work in the group and make record of all observations.
3. Students shall develop maintenance skill as expected by industries.
4. Student shall attempt to develop related hand-on skills and build confidence.
5. Student shall develop the habits of evolving more ideas, innovations, skills etc. apart from those included in scope of manual.
6. Student shall refer technical magazines and data books.
7. Student should develop a habit of submitting the experimentation work as per the schedule and s/he should be well prepared for the same.

General Guidelines for Software Engineering Laboratory Work

1. Student has to perform all the practical as described in the practical list.
2. For performing the practical list, student can able to work individually or make a team of maximum 3 students' group.
3. For making the group, students must take care that the all the students of group must be from same batch.
4. After establishing the team, every team will have to identify the problem area / definition for performing the laboratory work.
5. Every team has to approve their problem definition to respective faculty member within 15 days of the beginning of the semester.
6. Once the problem definition is approved by the faculty member, every team has to perform all the practical based on their respective problem definition.

Practical – 1

AIM: Define the Problem statement of Software development and after identifying the requirements based on the requirement engineering tasks prepare the Software Requirement Specification (SRS) document.

- **Objectives:** To learn how to define the problem statement for software development and how to identify the requirements while performing requirements engineering tasks and to learn what are the contents of SRS and how to write the contents in SRS.

- **Theory:**

  - The process of collecting the software requirement from the client then understand, evaluate and document it is called as requirement engineering.
  - Requirement engineering constructs a bridge for design and construction.

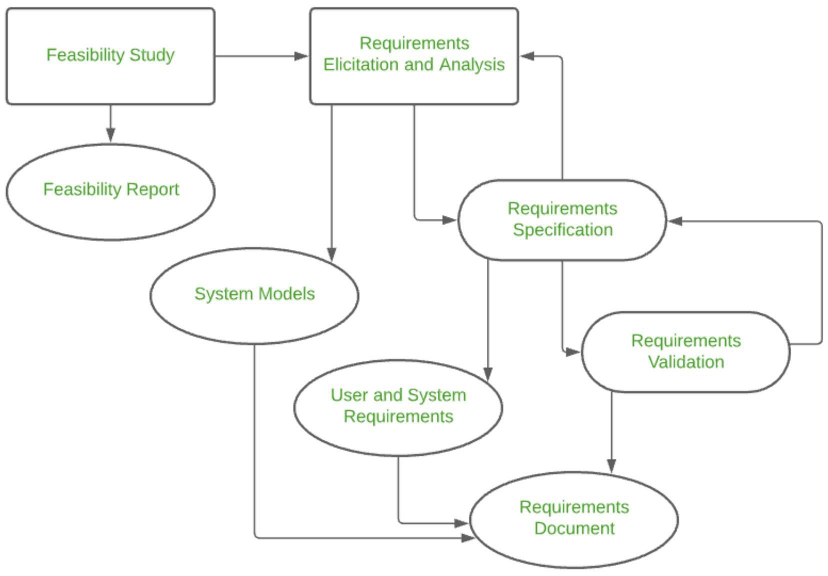


Figure: Requirements Engineering Process

- The production of the requirements stage of the software development process is **Software Requirements Specifications (SRS)** (also called a **requirements document**). This report lays a foundation for software engineering activities and is constructing when entire requirements are elicited and analyzed. **SRS** is a formal report, which acts as a representation of software that enables the customers to review whether it (SRS) is according to their requirements. Also, it comprises user requirements for a system as well as detailed specifications of the system requirements.

- The SRS is a specification for a specific software product, program, or set of applications that perform particular functions in a specific environment. It serves several goals depending on who is writing it. First, the SRS could be written by the client of a system. Second, the SRS could be written by a developer of the system. The two methods create entirely various situations and establish different purposes for the document altogether. The first case, SRS, is used to define the needs and expectation of the users. The second case, SRS, is written for various purposes and serves as a contract document between customer and developer.

- **Characteristics of good SRS**

1. **Correctness**
2. **Completeness**
3. **Consistency**
4. **Unambiguousness**
5. **Ranking for importance and stability**
6. **Modifiability**
7. **Verifiability**
8. **Traceability**
9. **Design Independence**
10. **Testability**
11. **Understandable by the customer**
12. **The right level of abstraction**

The following are the features of a good SRS document:

1. **Correctness:** User review is used to provide the accuracy of requirements stated in the SRS. SRS is said to be perfect if it covers all the needs that are truly expected from the system.
2. **Completeness:** The SRS is complete if, and only if, it includes the following elements:

   (1). All essential requirements, whether relating to functionality, performance, design, constraints, attributes, or external interfaces.

   (2). Definition of their responses of the software to all realizable classes of input data in all available categories of situations.

   (3). Full labels and references to all figures, tables, and diagrams in the SRS and definitions of all terms and units of measure.

3. **Consistency:** The SRS is consistent if, and only if, no subset of individual requirements described in its conflict. There are three types of possible conflict in the SRS:
   (1). The specified characteristics of real-world objects may conflicts. For example,
   (a) The format of an output report may be described in one requirement as tabular but in another as textual.
   (b) One condition may state that all lights shall be green while another states that all lights shall be blue.
   (2). There may be a reasonable or temporal conflict between the two specified actions. For example,
   (a) One requirement may determine that the program will add two inputs, and another determine that the program will multiply them.
   (b) One condition may state that "A" must always follow "B," while other requires that "A and B" co-occurs.
   (3). Two or more requirements may define the same real-world object but use different terms for that object. For example, a program's request for user input may be called a "prompt" in one requirement's and a "cue" in another. The use of standard terminology and descriptions promotes consistency.

4. **Unambiguousness:** SRS is unambiguous when every fixed requirement has only one interpretation. This suggests that each element is uniquely interpreted. In case there is a method used with multiple definitions, the requirements report should determine the implications in the SRS so that it is clear and simple to understand.
5. **Ranking for importance and stability:** The SRS is ranked for importance and stability if each requirement in it has an identifier to indicate either the significance or stability of that particular requirement.

   Typically, all requirements are not equally important. Some prerequisites may be essential, especially for life-critical applications, while others may be desirable. Each element should be identified to make these differences clear and explicit. Another way to rank requirements is to distinguish classes of items as essential, conditional, and optional.

6. **Modifiability:** SRS should be made as modifiable as likely and should be capable of quickly obtain changes to the system to some extent. Modifications should be perfectly indexed and cross-referenced.
7. **Verifiability:** SRS is correct when the specified requirements can be verified with a cost-effective system to check whether the final software meets those requirements. The requirements are verified with the help of reviews.

8. **Traceability:** The SRS is traceable if the origin of each of the requirements is clear and if it facilitates the referencing of each condition in future development or enhancement documentation.

   There are two types of Traceability:

   1. Backward Traceability: This depends upon each requirement explicitly referencing its source in earlier documents.

   2. Forward Traceability: This depends upon each element in the SRS having a unique name or reference number.

   The forward traceability of the SRS is especially crucial when the software product enters the operation and maintenance phase. As code and design document is modified, it is necessary to be able to ascertain the complete set of requirements that may be concerned by those modifications.

9. **Design Independence:** There should be an option to select from multiple design alternatives for the final system. More specifically, the SRS should not contain any implementation details.
10. **Testability:** An SRS should be written in such a method that it is simple to generate test cases and test plans from the report.
11. **Understandable by the customer:** An end user may be an expert in his/her explicit domain but might not be trained in computer science. Hence, the purpose of formal notations and symbols should be avoided too as much extent as possible. The language should be kept simple and clear.
12. **The right level of abstraction:** If the SRS is written for the requirements stage, the details should be explained explicitly. Whereas, for a feasibility study, fewer analysis can be used. Hence, the level of abstraction modifies according to the objective of the SRS.

Properties of a good SRS document

The essential properties of a good SRS document are the following:

Concise: The SRS report should be concise and at the same time, unambiguous, consistent, and complete. Verbose and irrelevant descriptions decrease readability and also increase error possibilities.

Structured: It should be well-structured. A well-structured document is simple to understand and modify. In practice, the SRS document undergoes several revisions to cope up with the user requirements. Often, user requirements evolve over a period of time. Therefore, to make the modifications to the SRS document easy, it is vital to make the report well-structured.

Black-box view: It should only define what the system should do and refrain from stating how to do these. This means that the SRS document should define the external behavior of the system and not discuss the implementation issues. The SRS report should view the system to be developed as a black box and should define the externally visible behavior of the system. For this reason, the SRS report is also known as the black-box specification of a system.

Conceptual integrity: It should show conceptual integrity so that the reader can merely understand it. Response to undesired events: It should characterize acceptable responses to unwanted events. These are called system responses to exceptional conditions.

Verifiable: All requirements of the system, as documented in the SRS document, should be correct. This means that it should be possible to decide whether or not requirements have been met in an implementation.

- **Background / Preparation:**

  Requirement engineering consists of seven different tasks as follows:

  1. Inception

  - Inception is a task where the requirement engineering asks a set of questions to establish a software process.
  - In this task, it understands the problem and evaluates with the proper solution.
  - It collaborates with the relationship between the customer and the developer.
  - The developer and customer decide the overall scope and the nature of the question.

  ✓ **Elicitation**
  Elicitation means to find the requirements from anybody.
  The requirements are difficult because the **following problems occur in elicitation**.
  Problem of scope: The customer give the unnecessary technical detail rather than clarity of the overall system objective.
  Problem of understanding: Poor understanding between the customer and the developer regarding various aspect of the project like capability, limitation of the computing environment.
  Problem of volatility: In this problem, the requirements change from time to time and it is difficult while developing the project.

- ✓ **Elaboration**

    - In this task, the information taken from user during inception and elaboration and are expanded and refined in elaboration.
    - Its main task is developing pure model of software using functions, feature and constraints of a software.

4. Negotiation

- In negotiation task, a software engineer decides the how will the project be achieved with limited business resources.
- To create rough guesses of development and access the impact of the requirement on the project cost and delivery time.

5. Specification

- In this task, the requirement engineer constructs a final work product.
- The work product is in the form of software requirement specification.
- In this task, formalize the requirement of the proposed software such as informative, functional and behavioral.
- The requirement are formalize in both graphical and textual formats.

6. Validation

- The work product is built as an output of the requirement engineering and that is accessed for the quality through a validation step.
- The formal technical reviews from the software engineer, customer and other stakeholders helps for the primary requirements validation mechanism.

7. Requirement management

- It is a set of activities that help the project team to identify, control and track the requirements and changes can be made to the requirements at any time of the ongoing project.
- These tasks start with the identification and assign a unique identifier to each of the requirement.
- After finalizing the requirement traceability table is developed.
- The examples of traceability table are the features, sources, dependencies, subsystems and interface of the requirement.

- In order to form a good SRS, here you will see some points which can be used and should be considered to form a structure of good SRS. These are as follows :

  1. Introduction
     - **(i)** Purpose of this document
     - **(ii)** Scope of this document
     - **(iii)** Overview
  2. General description
  3. Functional Requirements
  4. Interface Requirements
  5. Performance Requirements
  6. Design Constraints
  7. Non-Functional Attributes
  8. Preliminary Schedule and Budget
  9. Appendices

- **Tools / Material Needed:**
  - **Hardware:**
  - **Software:**

- **Procedure:**
  - Every team has to identify the requirements of their projects and arrange as per the requirement engineering document.

- **Steps:**
  - Every team has to follow the given format as described in the below example.
  - Example 1: **Withdraw Cash from ATM**

R.1: withdraw cash	
Description:	The withdraw cash function first determines the type of account that the user has and the account number from which the user wishes to withdraw cash. It checks the balance to determine whether the requested amount is available in the account. If enough balance is available, it outputs the required cash, otherwise it generates an error message.
R.1.1: Select withdraw amount option	
Input:	“withdraw amount” option
Output:	user prompted to enter the account type
R.1.2: Select account type	
Input:	user option
Output:	prompt to enter amount
R.1.3: Get required amount	
Input:	amount to be withdrawn in integer values greater than 100 and less than 10,000 in multiples of 100.
Output:	The requested cash and printed transaction statement.
Processing:	The amount is debited from the user’s account if sufficient balance is available, otherwise an error message displayed.

- Example 2: **Search Book Availability in Library**

R.1: Search Book	
Description:	Once the user selects the search option, he would be asked to enter the keywords. The system would search the book in the book list based on the keywords entered. After making the search, the system should output the details of all books whose title or author name match any of the keywords entered. The book details to be displayed include: title, author name, publisher name, year of publication, ISBN number, catalog number, and the location in the library.
R.1.1: Select search option	
Input:	“Search” option
Output:	User prompted to enter the keywords
R.1.2: Search and display	
Input:	Keywords
Output:	Details of all books whose title or author name matches any of the keywords

	entered by the user. The details displayed would include: title of the book, author name, ISBN number, catalog number, year of publication, number of copies available, and the location in the library.
Processing:	Search the book list based on the keywords
R.2: Renew book	
Description:	When the "renew" option is selected, the user is asked to enter his membership number and password. After password validation, the list of the books borrowed by him are displayed. The user can renew any of his borrowed books by indication them. A requested book cannot be renewed if it is reserved by another user. In this case, an error message would be displayed.
R.2.1: Select renew option	
State:	The user has logged in and the main menu has been displayed
Input:	"Renew" option selection
Output:	Prompt message to the user to enter his membership number and password
R.2.2: Login	
State:	The renew option has been selected
Input:	Membership number and password
Output:	List of the books borrowed by the user is displayed, and user is prompted to select the books to be renewed, if the password is valid. If the password is invalid, the user is asked to reenter the password.
Processing:	Password validation search the books issued to the user form the borrower's list and display
Next function:	**R.2.3 if password is valid and R.2.2 if password is invalid**
R.2.3: Renew selected books	
Input:	User choice for books to be renewed out of the books borrowed by him.
Output:	Confirmation of the books successfully renewed and apology message for the books that could not be renewed.
Processing:	Check if anyone has reserved any of the requested books. Renew the books selected by the use in the borrower's list, if no one has reserved those books.

Software Requirement Specification (SRS) Format as name suggests, is complete specification and description of requirements of software that needs to be fulfilled for successful development of software system. These requirements can be functional as well as non-functional depending upon type of requirement. The interaction between different customers and contractor is done because its necessary to fully understand needs of customers.

Document Title
Author(s)
Affiliation
Address
Date
Document Version

Depending upon information gathered after interaction, SRS is developed which describes requirements of software that may include changes and modifications that is needed to be done to increase quality of product and to satisfy customer's demand.

1. **Introduction :**

    - **(i) Purpose of this Document –**
        - At first, main aim of why this document is necessary and what's purpose of document is explained and described.
    - **(ii) Scope of this document –**
        - In this, overall working and main objective of document and what value it will provide to customer is described and explained. It also includes a description of development cost and time required.
    - **(iii) Overview –**
        - In this, description of product is explained. It's simply summary or overall review of product.

2. **General description :**
    - In this, general functions of product which includes objective of user, a user characteristic, features, benefits, about why its importance is mentioned. It also describes features of user community.
3. **Functional Requirements:**
    - In this, possible outcome of software system which includes effects due to operation of program is fully explained. All functional requirements which may include calculations, data processing, etc. are placed in a ranked order.
4. **Interface Requirements :**
    - In this, software interfaces which mean how software program communicates with each other or users either in form of any language, code, or message are fully described and explained. Examples can be shared memory, data streams, etc.
5. **Performance Requirements:**
    - In this, how a software system performs desired functions under specific condition is explained. It also explains required time, required memory, maximum error rate, etc.

6. **Design Constraints:**
    - In this, constraints which simply means limitation or restriction are specified and explained for design team. Examples may include use of a particular algorithm, hardware and software limitations, etc.
7. **Non-Functional Attributes:**
    - In this, non-functional attributes are explained that are required by software system for better performance. An example may include Security, Portability, Reliability, Reusability, Application compatibility, Data integrity, Scalability capacity, etc.
8. **Preliminary Schedule and Budget:**
    - In this, initial version and budget of project plan are explained which include overall time duration required and overall cost required for development of project.
9. **Appendices:**
    - In this, additional information like references from where information is gathered, definitions of some specific terms, acronyms, abbreviations, etc. are given and explained.

Quiz:

1. Which are properties of good SRS?
2. What is functional and non-functional requirement?
3. Which are different requirement engineering tasks?

Suggested Reference:

1. Lecture on "System Analysis and Design", NPTEL
2. When Telepathy Won't Do: Requirements Engineering Key Practices
3. Requirements Analysis: Process of requirements gathering and requirement definition
4. IEEE Recommended Practice for Software Requirements Specifications
5. Requirements Trace-ability and Use Cases

SOFTWARE REQUIREMENT SPECIFICATION FOR “HOTEL MANAGEMENT SYSTEM”

Authors:

1. Mohit Rajendra Mahajan
2. Parth Kishorbhai Mandaviya

Institute:	Government Engineering College, Modasa
Department:	Computer Engineering
Date:	28th Aug, 2023
Version:	1.0

1. Introduction:

- **Purpose of this Document:**
  General purpose of this document is to specify the Hotel Management System (HMS) in detail, with it’s functional and non-functional requirements. At the end the project developers are going to implement the following mentioned requirement of the project (mentioned in this document).

- **Scope of this Document:**
  The scope of this document is to provide to essential and detailed information about the project to the reader. It will discuss over the chef specification, vehicles parking preference, room service for the guests, table and menu specification for the guest, providing special medical facilities.

- **Overview:**
  There are already numerous Hotel Management System available but as we observe there are some problems on which we are not paying attention so we are getting those point in consideration and working on it to develop the current Hotel Management Systems.

2. General description:

In today’s fast-paced hospitality industry, effective management of hotel operations is crucial to ensure a seamless and satisfying experience for guests. The Hotel Management System (HMS) project is designed to address the various challenges faced by hotel owners, managers, and staff in managing their day-to-day operations efficiency and providing top-notch services to guests.

- **Key Features: -**
  a) We specify the live parking system for our guests to ensure their vehicle’s parking in the parking space (provided by the Hotel).
  b) In our project they get a unique facility for their food or meal as they can choose their meal’s chief for their particular dish.
  c) We also provide an option to the guest to book the specified room services as their needs.

- **Project Function: -**
  a) Live Parking
  b) Hotel room booking (with specified staff)
  c) Hotel menu
  d) Table booking
  e) Chief assigning
  f) Payment processing
  g) Special offers
  h) Multi-language and currency support
  i) Guest review and rating

j) Report and analytics
k) Customer support and chat
l) Social media integration
m) Weather and local information
n) Security and compliance (the act of obeying a law or rule)
o) Feedback and surveys
p) Check-in and check-out

- **Benefits to the users: -**

They are not supposed to wait for their meals due to our menu option. In this option they can order their meal before they reach the hotel or they report their reaching time according to guest's schedule and hotel will arrange their meal by that time for them.

The live parking system will give all information about the parking space of the hotel so the guest will prefer the hotel if there is the space available for the parking, this will avoid the unnecessary chaos in the parking space.

The hotel room booking with the new function as specific staff will help the guests to make their more comfortable as they want.

For example: Your grandmother in the new town and you want some experienced and skilled staff should be serving her. So here she will be more comfortable with the lady staff as compare to the male staff so you can make choice for this facility in our project.

The most important benefit is the meal's chief specification and this will increase the comfort level of the guest to the next level.

In our project we are targeting the guest's comfort and time. With this we also ensure our user should get best using experience with Hotel Management System.

3. Function Requirements:

R.1: User Authentication	
Description	User registration and login for guests, staff and administrators.
R.1.1: Sign-in and registration	
Input	Users are supposed to enter Username, phone, e-mail, password, any gov. id etc.
Output	User gets a new page pop up for login procedure.
R.1.2: Login	
Input	User are supposed to enter the username and password to login the site.
Output	User will enter the main page of the site where they get menu, booking, profile etc.

R.2: Booking and Reservation	
Description	User's room booking and availability status. Selection of room types, dates, and guests. Live tracking of parking for getting the live update of the parking areas.
R.2.1: Room Booking	
Input	User select the room type, floor, services, date of reservation, no of guests, contact details etc.
Output	System will show the no of rooms occupied and not occupied according to user choice.
R.2.2: Parking Booking	
Input	User enter their Vehicle details and parking position.
Output	System will show how many parking slots are available and book the user's choice slot.
R.2.3: Table Reservation	
Input	User enter the no of guests, user's meal, name for the reservation, table no, contact etc.
Output	System will show the available tables, menu, show offers, assigned table no (if random), user's meal (when user arrived).

R.3: Room Checking details	
Description	Online check-in and check-out for users with payment processing
R3.1: Room check-in	
Input	User enter the room details, and opt send to the user for verification.
Output	System will verify the user's otp and then note the check-in time in the database.
R.3.2: Room Check-out	
Input	User enter just chose the check-out option given to the page.
Output	System will note the time of check-out and then room service will verify the room by offline visit.
R.3.3: Room billing	
Input	When User is booking room and other services then user will get the pop-up for payment procedure, where the user will go for UPI, debit card, credit card, gift card (if available) etc.
Output	User will receive the bill of the payment and the system will register the entry of the user detail about the reservation in the database.

R.4: Weather and local information	
Description	System will provide the weather forecasts and local attractions/event information for the tourists and guests to visit the hotel for getting rest.
R.4.1: Local weather report	
Input	User enter the current city or arrival time in the city.
Output	System will analysis the weather around user and show the best time to visit the hotel.
R.4.2: Local area information	
Input	User enter the current city or the purpose of visit (so the system will suggest best places to visit).
Output	System will show the best places and time to visit the spots according to your needs.

R.5: Social Media Integration	
Description	System has a option to integrate the social media account with the user's id.
R.5.1: Social marketing	
Input	User will enter the account's username and the platform where's it is available.
Output	System will share their experience about the staying on hotel's social media accounts with the user's permission. User are also allowed to post details about the hotel and their experiences about the stay.

R.6: Gift cards and vouchers	
Description	System will show this option whenever the user is purchasing something.
R.6.1: Apply the gift cards while payment or billing	
Input	While the payment or billing in procedure then user can choose the offer.
Output	System will apply the gift card and show the reduced amount and proceed for the payment.

R.7: Notifications and alert	
Description	System will send Notification to the guests regarding gift cards, offers, checking updates, check-in and check-out remainders, reservation updates etc.
R.7.1: Notifications	
Output	System will provide the soft copy of the Bills, Gift cards, offers, checking updates, etc.
R.7.2: Alerts	
Output	Check-in and check-out remainders, reservation details (if full) etc.

R.8: Review and Rating	
Description	System will take the users review after the check-out procedure and note the review for further updates in system.
R.8.1: Rating and Review	
Input	User enter the name and their experience about the stay in the hotel.
Output	System will encourage guests to visit the hotel again.

4. Interface Requirements:

- **User Interface:**

a) Login/Registration:

User-friendly login and registration form.
Password reset and recovery options.

b) Dashboard:

Intuitive and visually appealing dashboard for administrators.
Display of relevant information such as occupancy rates, revenue, and pending tasks.

c) Room management:

Interface for adding, editing, or deleting rooms.

Room status indicators (clean, occupied, available) with color coding.
Detailed room information with photos and amenities.

d) Check-in/Check-out Interface:

Streamlined check-in and check-out processes.
Options to add guest details, ID verification, and payment processing.
Electronic signature capture for check-in.

e) Payment Interface:

Secure payment gateway integration.
Options for entering credit card details or choosing other payment methods.
Clear billing breakdown and invoice generation.

f) Guest Services Interface:

Easy-to-use interface for ordering additional services (room service, housekeeping, etc.).

g) Reservation Management Interface:

Interface for modifying or canceling reservations.
Access to reservation history and details.

h) Admin Control Panel:

Admin panel with menus and options for managing users, rooms, staff, and reports.

i) Reporting and Analytics:

Interface for generating reports and visualizing analytics data.

- **Hardware Interfaces:**
  Ensure compatibility with various hardware devices, including desktop computers, laptops, tablets, and smartphones.
  Integration with card readers and electronic signature capture devices for check-in and payment processing.
- **Software Interfaces:**
  Integration with third-party software and services, such as payment gateways, property management systems, and channel managers.
- **Communication Interfaces:**
  Ensure that the system can communicate with external systems through APIs or web services.
  Support for email notifications and alerts to users and administrators.
- **Accessibility Interfaces:**
  Implement accessibility features for users with disabilities, including screen readers, keyboard navigation, and text-to-speech functionality.
- **Internationalization and Localization:**
  Support multiple languages and currencies to cater to international guests.
  Implement date and time formats, numbering systems, and cultural considerations based on the user's location.
- **Security Interfaces:**
  Implement secure authentication and authorization mechanisms.

Integration with security protocols and encryption standards to protect user data.

- **Social Media Integration:**
  Social media sharing buttons and interfaces for guests to share their experiences.
- **API Documentation:**
  Provide clear documentation for any APIs that the HMS offers for third-party integration.

5. Performance requirement:

- **Response Time:**
  The system should respond to user interactions (e.g., booking, room status updates) within a specified time frame, such as:
  95% of requests must be processed within 2 seconds.
  Check-in and check-out operations should complete in under 5 minutes.
- **System Availability:**
  The HMS should be available 24/7, with minimal downtime for maintenance or upgrades.
  Define an acceptable uptime percentage (e.g., 99.9% availability) for the system.
- **Scalability:**
  The system should be able to handle increased load during peak booking periods (e.g., holidays).
  Define how many concurrent users or transactions the system should support without performance degradation.
  Implement load balancing and auto-scaling mechanisms to distribute traffic efficiently.
- **Resource Utilization:**
  Monitor and optimize CPU, memory, and storage utilization to ensure efficient resource management.
  Set limits on resource usage to prevent resource exhaustion.
- **Database Performance:**
  Database queries, especially those related to room availability and pricing, should execute quickly.
  Implement database indexing and caching mechanisms for faster data retrieval.
  Define acceptable database response times for various operations (e.g., room search, reservation retrieval).
- **Network Performance:**
  Ensure fast and reliable communication between the HMS and external systems (e.g., payment gateways, third-party integrations).
  Define acceptable network latency and bandwidth requirements.
- **Load Testing:**
  Conduct load testing to simulate high levels of concurrent user activity.
  Identify performance bottlenecks and optimize system components accordingly.
  Define load testing scenarios, such as peak booking periods or simultaneous check-ins/check-outs.
- **Security Performance:**

Ensure that security measures (e.g., encryption, authentication) do not significantly impact system performance.
Regularly assess the system's ability to withstand security threats without degrading performance.

- **Reporting Performance:**

  Reports and analytics should generate within an acceptable time frame, even for extensive datasets.

  Specify maximum report generation times for different report types.
- **Third-Party Integrations:**

  External services and APIs should respond promptly to requests from the HMS.
  Define acceptable response times for third-party integrations.
- **Caching and Content Delivery:**

  Utilize caching mechanisms to reduce server load and accelerate content delivery.
  Implement Content Delivery Networks (CDNs) for static assets to improve load times.
- **Error Handling and Logging:**

  Efficiently handle errors and exceptions without causing system slowdowns.
  Log performance-related data for monitoring and troubleshooting.
- **Mobile Responsiveness:**

  Ensure that mobile interfaces are responsive and provide acceptable load times on mobile devices.
- **Compliance with Industry Standards:**

  Adhere to industry standards and best practices for performance optimization.
- **Regular Performance Testing and Monitoring:**

  Establish a schedule for ongoing performance testing and monitoring to identify and address performance issues proactively.
- **User Experience:**

  Measure and optimize user experience by considering factors like page load times and responsiveness.

6. Design Constraints:

- **Technology Constraints:**

  Legacy Systems: Integration with existing legacy systems may impose constraints on the choice of technology and architecture.
  Platform: The HMS may need to be developed for specific platforms or operating systems (e.g., Windows, iOS, Android).
- **Budget constraints:**

  Cost Limitations: There may be budget limitations that affect the choice of technologies, hardware, and software licenses.
  Resource Availability: Constraints on the availability of financial and human resources can impact system development.

- **Localization and Globalization Constraints:**

  Multilingual Support: The system may need to support multiple languages and currencies, which can affect user interface design and database structure.
  Cultural Considerations: Cultural differences in user behavior and expectations may influence the system's design and features.
- **Security Constraints:**

  Security Policies: Compliance with the hotel's security policies and standards, which may affect system design and access control.
- **Data Constraints:**

  Data Volume: Constraints related to the volume of data to be stored and processed, which can affect database design and storage solutions.
- **Backup and Recovery Constraints:**

  Backup Policies: Constraints related to data backup and disaster recovery procedures.

7. Non-Functional Attributes:

- **Performance:**

  Response Time: The system should respond to user interactions quickly, with defined maximum response times for various operations (e.g., room booking, check-in).
  Throughput: The system should handle a specified number of concurrent users or transactions per second.
  Scalability: The system should be scalable to accommodate increased load during peak periods.
- **Security:**

  Data Security: Data should be protected through encryption, access control, and secure authentication mechanisms.
  Compliance: The system should comply with data protection regulations (e.g., GDPR) and industry security standards.
  Authentication and Authorization: Strong user authentication and authorization controls should be in place.
  Audit Trails: Logging and auditing of system activities should be implemented for security monitoring.
- **Documentation and Training:**

  Comprehensive documentation for users and administrators.
  Training materials and user guides to ensure effective system utilization.
- **Usability and User Experience (UX):**

  User-Friendly Interface: The system should have an intuitive and user-friendly interface.
  Accessibility: The system should be accessible to individuals with disabilities, adhering to accessibility standards (e.g., WCAG).
  Consistency: User interface elements and interactions should be consistent throughout the application.
- **Compatibility:**

  The system should be compatible with various web browsers, operating systems, and devices (desktop, mobile, tablet).

- **Availability:**

  Uptime: The system should be available 24/7, with minimal planned downtime for maintenance.
  Fault Tolerance: The system should continue to operate in the presence of hardware or software failures.
- **Reliability:**

  The system should consistently perform its intended functions without errors or failures.
  Mean Time Between Failures (MTBF) and Mean Time to Repair (MTTR) should be defined and optimized.

8. Appendices:

1) Google Translate
   www.translate.google.co.in

2) PyPI · The Python Package Index
   www.pypi.org

3) Python Dictionaries (w3schools.com)
   www.w3schools.com

4) The web framework for perfectionists with deadlines | Django (djangoproject.com)
   www.djangoproject.com

5) Bootstrap · The most popular HTML, CSS, and JS library in the world. (getbootstrap.com)
   getbootstrap.com

6) ORM Quick Start — SQLAlchemy 2.0 Documentation
   docs.sqlalchemy.org

Tool/Material needed:

Hardware Requirement:

Smart Phone device.
Display with resolutions 480x854 pixel.
Processor: Quad-core 1.1 GHz or higher.
Built-in GPS for location tracking.

Software Requirement:

Any browser: HTML5 supported, JS enabled.
Database at server side.

Android device:

O.S: Android 5.0 or higher.

iOS device:

O.S: iOS 11 or highe

Rubric wise marks obtained:

Rubrics	1	2	3	4	5	Total
Marks	Complete implementation as asked	Complete implementation as asked Problem analysis	Complete implementation as asked Problem analysis Development of the Solution	Complete implementation as asked Problem analysis Development of the Solution Concept Clarity & understanding	Complete implementation as asked Problem analysis Development of the Solution Concept Clarity & understanding Correct answer to all questions	

Signature of Faculty:

Practical – 2

AIM: Prepare the Software Project Management Plan (SPMP) including following: 1) Estimation of Size, Cost, Duration, Effort 2) Prepare the Schedule, Milestones using Gantt chart

- **Objectives:**

1. To perform Estimation of Size, Cost, Duration, Effort
2. To Prepare the Schedule, Milestones

- **Theory:**

Once a project is found to be feasible, software project managers undertake project planning. Project planning is undertaken and completed even before any development activity starts. Project planning consists of the following essential activities:

Estimating the following attributes of the project:

Project size: What will be problem complexity in terms of the effort and time required to develop the product?

Cost: How much is it going to cost to develop the project?

Duration: How long is it going to take to complete development?

Effort: How much effort would be required?

The effectiveness of the subsequent planning activities is based on the accuracy of these estimations.

- Scheduling manpower and other resources.
- Staff organization and staffing plans.
- Risk identification, analysis, and abatement planning
- Miscellaneous plans such as quality assurance plan, configuration management plan, etc.

Organization of SPMP Document

Introduction (Objectives, Major Functions, Performance Issues, Management and Technical Constraints)

Project Estimates (Historical Data, Estimation Techniques, Effort, Cost, and Project Duration Estimates)

Project Resources Plan(People, Hardware and Software, Special Resources)

Schedules (Work Breakdown Structure, Task Network, Gantt Chart Representation, PERT Chart Representation)

Risk Management Plan (Risk Analysis, Risk Identification, Risk Estimation, Abatement Procedures)

Project Tracking and Control Plan

Miscellaneous Plans(Process Tailoring, Quality Assurance)
Function Points

STEP 1: measure size in terms of the amount of functionality in a system. Function points are computed by first calculating an ***unadjusted function point count*** **(UFC).**
Counts are made for the following categories
External inputs–those items provided by the user that describe distinct application-oriented data (such as file names and menu selections)

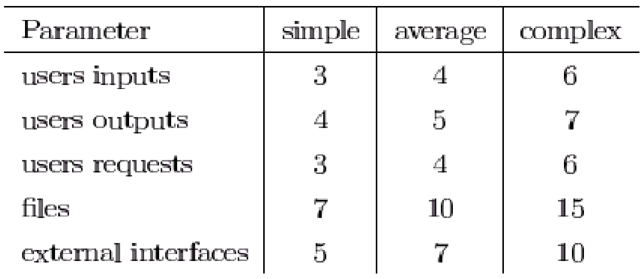
External outputs–those items provided to the user that generate distinct application-oriented data (such as reports and messages, rather than the individual components of these)

External inquiries–interactive inputs requiring a response

External files–machine-readable interfaces to other systems

Internal files–logical master files in the system

STEP 2: Multiply each number by a weight factor, according to complexity (**simple**, **average** or **complex**) of the parameter, associated with that number. The value is given by a table:



Parameter	simple	average	complex
users inputs	3	4	6
users outputs	4	5	7
users requests	3	4	6
files	7	10	15
external interfaces	5	7	10

STEP 3: Calculate the total **UFP**(Unadjusted Function Points)

STEP 4: Calculate the total **TCF**(Technical Complexity Factor) by giving a value between 0 and 5 according to the importance of the following points(next slide):
Technical Complexity Factors:
1.Data Communication 2.Distributed Data Processing 3.Performance Criteria 4.Heavily Utilized Hardware 5.High Transaction Rates 6.Online Data Entry 7.Online Updating 8.End-user Efficiency 9.Complex Computations 10.Reusability 11.Ease of Installation 12.Ease of Operation 13.Portability 14.Maintainability

STEP 5: Sum the resulting numbers to obtain **DI**(degree of influence)

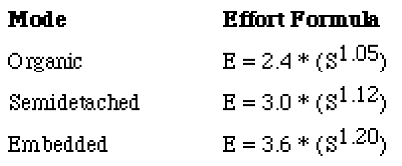
STEP 6: **TCF**(Technical Complexity Factor) by given by the formula
–***TCF=0.65+0.01*DI***
STEP 6: Function Points are by given by the formula
–***FP=UFP*TCF***

The **Constructive Cost Model** (COCOMO) is the most widely used software estimation model.

The COCOMO model predicts the **effort** and **duration** of a project based on inputs relating to the size of the resulting systems and a number of "**cost drives**" that affect productivity.
The most important factors contributing to a project's duration and cost is the Development Mode
Organic Mode: The project is developed in a familiar, stable environment, and the product is similar to previously developed products. The product is relatively small, and requires little innovation.

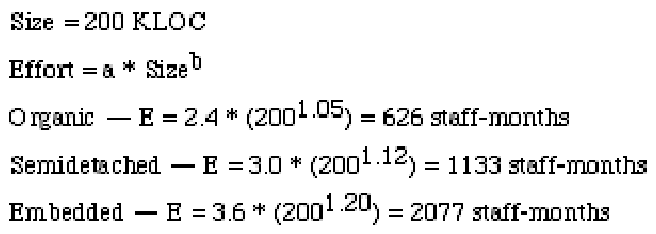
Semidetached Mode: The project's characteristics are intermediate between Organic and Embedded.

Embedded Mode: The project is characterized by tight, inflexible constraints and interface requirements. An embedded mode project will require a great deal of innovation.



Mode	**Effort Formula**
Organic	$E = 2.4 * (S^{1.05})$
Semidetached	$E = 3.0 * (S^{1.12})$
Embedded	$E = 3.6 * (S^{1.20})$

Example:



Size = 200 KLOC

Effort $= a * \text{Size}^{b}$

Organic — **E** $= 2.4 * (200^{1.05}) =$ 626 staff-months

Semidetached — **E** $= 3.0 * (200^{1.12}) =$ 1133 staff-months

Embedded — E $= 3.6 * (200^{1.20}) =$ 2077 staff-months

Project-task scheduling is an important project planning activity. It involves deciding which tasks would be taken up when. In order to schedule the project activities, a software project manager needs to do the following:

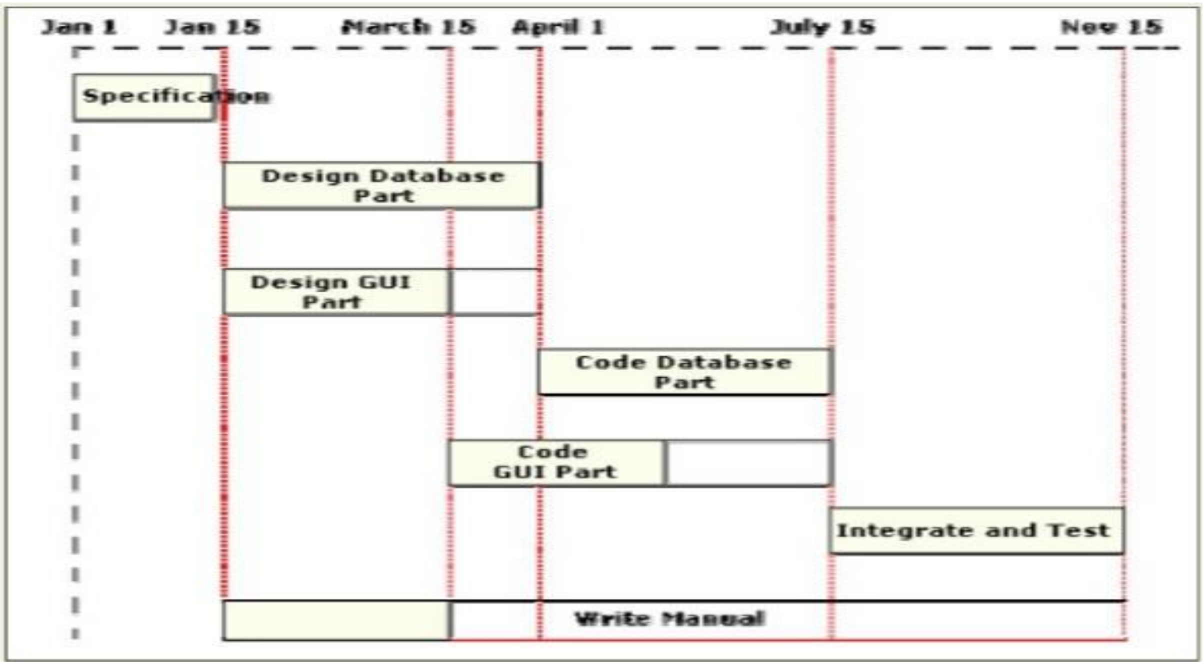
1. Identify all the tasks needed to complete the project.
2. Break down large tasks into small activities.
3. Determine the dependency among different activities.
4. Establish the most likely estimates for the time durations necessary to complete the activities.
5. Allocate resources to activities.
6. Plan the starting and ending dates for various activities.
7. Determine the critical path.

A critical path is the chain of activities that determines the duration of the project. The first step in scheduling a software project involves identifying all the tasks necessary to complete the project. A good knowledge of the intricacies of the project and the development process helps the managers to effectively identify the important tasks of the project. Next, the large tasks are broken down into a logical set of small activities which would be assigned to different engineers. The work breakdown structure formalism helps the manager to breakdown the tasks systematically after the project manager has broken down the tasks and created the work breakdown structure, he has to find the dependency among the activities.

Dependency among the different activities determines the order in which the different activities would be carried out. If an activity A requires the results of another activity B, then activity A must be scheduled after activity B. In general, the task dependencies define a partial ordering among tasks, i.e. each tasks may precede a subset of other tasks, but some tasks might not have any precedence ordering defined between them (called concurrent task). The dependency among the activities is represented in the form of an activity network. Once the activity network representation has been worked out, resources are allocated to each activity.

Resource allocation is typically done using a Gantt chart. After resource allocation is done, a PERT chart representation is developed. The PERT chart representation is suitable for program monitoring and control. For task scheduling, the project manager needs to decompose the project tasks into a set of activities. The time frame when each activity is to be performed is to be determined. The end of each activity is called milestone. The project manager tracks the progress of a project by monitoring the timely completion of the milestones. If he observes that the milestones start getting delayed, then he has to carefully control the activities, so that the overall deadline can still be met.

Gantt charts are mainly used to allocate resources to activities. The resources allocated to activities include staff, hardware, and software. Gantt charts (named after its developer Henry Gantt) are useful for resource planning. A Gantt chart is a special type of bar chart where each bar represents an activity. The bars are drawn along a time line. The length of each bar is proportional to the duration of time planned for the corresponding activity. Gantt charts are used in software project management are actually an enhanced version of the standard Gantt charts. In the Gantt charts used for software project management, each bar consists of a white part and a shaded part. The shaded part of the bar shows the length of time each task is estimated to take. The white part shows the slack time, that is, the latest time by which a task must be finished.



Quiz:

1) Explain project scheduling process. Explain Gantt Chart in detail.
2) A project size of 300 KLOC is to be developed. Software development team has beginner experience on similar type of projects. The project schedule is not very tight. Calculate the Effort, development time, average staff size, and productivity of the project.
3) Explain Software metrics used for software cost estimation

Suggested Reference:

I. COCOMO
II. Halstead complexity measures
III. COCOMO (Constructive Cost Model), Seminar on Software Cost Estimation WS 2002 / 2003, presented by Nancy Merlo – Schett
IV. The Halstead metrics
V. Software Engineering, National Program on Technology Enhanced Learning
VI. Halstead Metrics, Verifysoft Technology

Organization of SPMP Document

Introduction:

The project, Hotel Management System is a web-based application that allows the hotel manager to handle all hotel activities online. The hotel manager is a very busy person and does not have the time to sit and manage the entire activities manually on paper. This application gives him the power and flexibility to manage the entire system from a single online system. Hotel management project provides room booking and other necessary hotel management features. Customers can view and book room online. The system is hence useful for both customers and managers to portable manage the hotel activities.

Project Estimation Techniques

- Function Points Analysis (FPA)
- COCOMO (Constructive Cost Model)

Function Points

FPA is an estimation technique that assesses the functionality provided by a software application from the user's perspective. It quantifies the functionality in terms of function points, which are categorized into different types, including external inputs, external outputs, external inquiries, internal logical files, and external interface files. FPA helps in estimating the size and complexity of a software project based on the identified functions and their complexity.

- **Step 1: count of all information domain:**

External inputs:	27
External outputs:	7
External inquiries:	17
External files:	3
Internal files:	6

- **Step 2: Calculate the total UFP (Unadjusted Function Points)**

Measurement Parameter	**Count**		**Weighting Factor Average**	
External inputs	27	*	4	108
External outputs	7	*	5	35
External inquiries	17	*	4	68
External files	3	*	10	30
Internal files	6	*	7	42
			Total Count:	283

- **Step 3: Calculate the total VAF (Value Adjustment Factor) by giving a value between 0 and 5 according to the importance of the following:**

1. Data Communication	3
2. Distributed Data Processing	4
3. Performance Criteria	4
4. Heavily Utilized Hardware	2.5
5. High Transaction Rates	5
6. Online Data Entry	2
7. Online Updating	2.5
8. End user Efficiency	4
9. Complex Computations	3
10. Reusability	3
11. Ease of Installation	4.5
12. Ease of Operation	4.5
13. Portability	4
14. Maintainability	5

Total Degree of influence (TDI): 51
Value Adjustment Factor (VAF) = 0.65 + 0.01 * TDI
= 0.65 + 0.01*51
VAF = 1.16

- **Step 4: calculate FP using Below Formula:**

FP = UFP * VAF
= UFP * (0.65 + 0.01 * TDI)
= 283 * 1.16
= 328.28

COCOMO MODEL

COCOMO is a family of estimation models, including Basic COCOMO and Intermediate COCOMO, which help in estimating the effort, cost, and duration of a software project. COCOMO models consider various factors, such as project size, complexity, development environment, and team experience.

While Function Points Analysis estimates the size of the software, COCOMO estimates effort and cost based on a combination of size and other project-specific factors.

Codes of line (COL): 370 KLOC

Organic	E=2.04 * (COL)^1.05
Semi-detached	E=3.00 * (COL)^1.12
embedded	E=3.60 * (COL)^1.20

- Effort (Semi-detached):

  $E = 3.00 * (COL)^{1.12}$

  $E = 3.00 * (370)^{1.12}$

  $E = 3.00 * (752.29)$

  E = 2256.87 Person month

- Duration (Semi-detached):

  $D = 2.5 * (E)^{0.35}$

  $D = 2.5 * (2256.87)^{0.35}$

  D = 37.30 Months

- Average staff size (Semi-detached):

  S = E/D

  S = 2256.87 /37.30

  S = 60.50 person

- Productivity:

  P = Codes of line (COL) / Average staff size

  P = 370 / 60.50

  P = 6.1157024793

- Calculate Cost:

  Duration = 37.30 Months

  Working hour = 8 Hrs

  Total working hour in month = 37.30 * 8 * 25 = 7460

  Cost per hour = 75

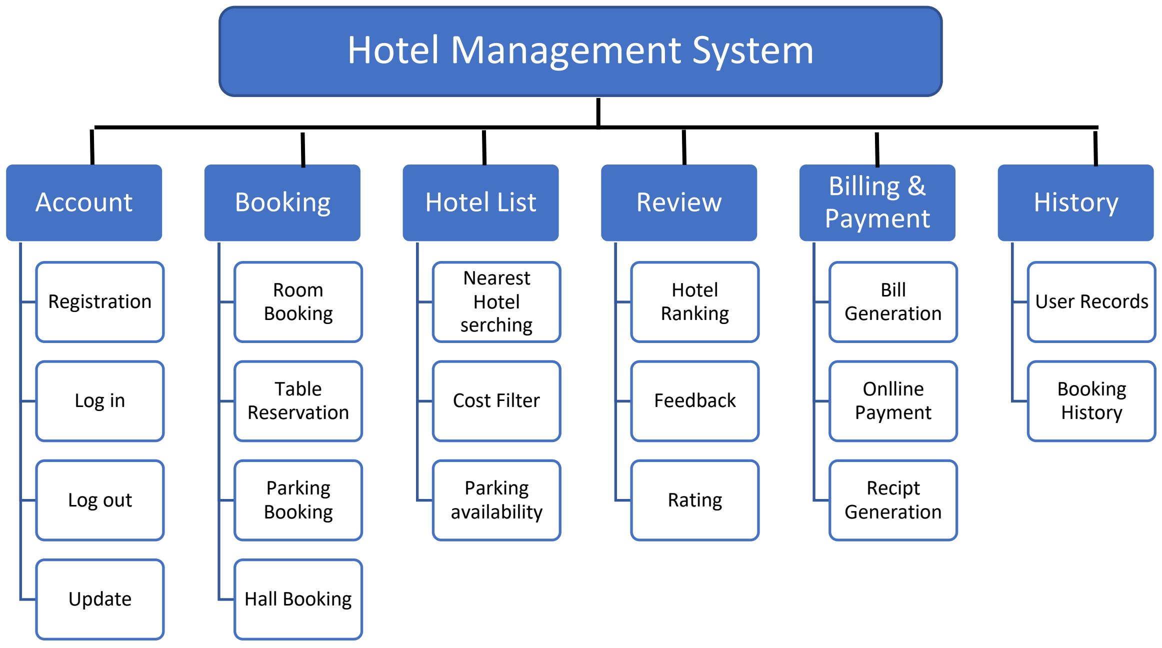
  Total cost =7460 * 75 = 559500 INR

Schedules

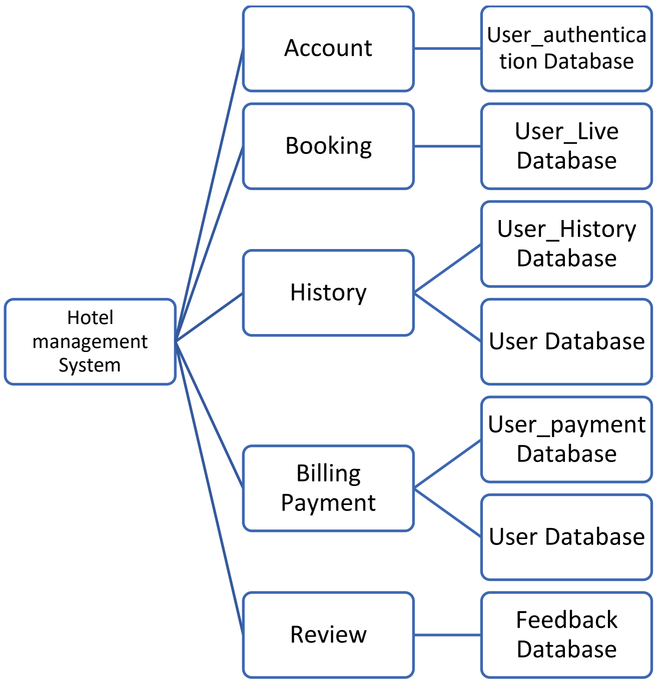
1. Identify all the tasks needed to complete the project.

- Booking
- Parking
- Registration
- Log in
- Log out
- Table reservation
- Weather Report
- Update user profile
- Gift cards and Voucher
- Notifications and alert
- Review and rating

2. Break down large tasks into small activities.



3. Determine the dependency among different activities.



4. Establish the most likely estimates for the time durations necessary to complete the activities.

Activities	Duration (in months)
Account	3
Booking	6
History	4
Billing & Payment	7
Review	4

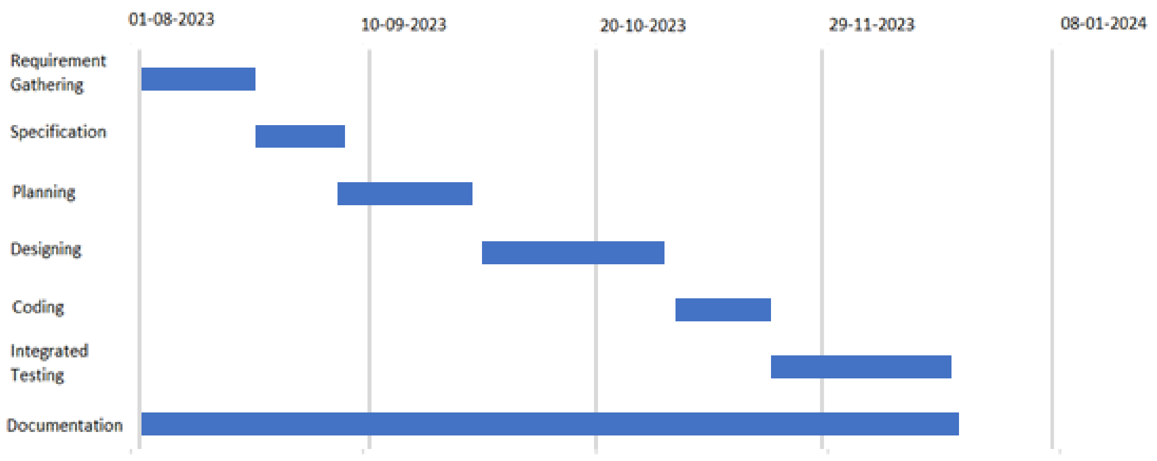
5. Allocate resources to activities.

Hotel Management System:

a. Account:
   i. User_authentication Database
b. Booking:
   i. User_Live Database
c. History:
   i. User Database
   ii. User_History Database
d. Billing & Payment:
   i. User_payment Database
   ii. User Database
e. Review:
   i. Feedback Database

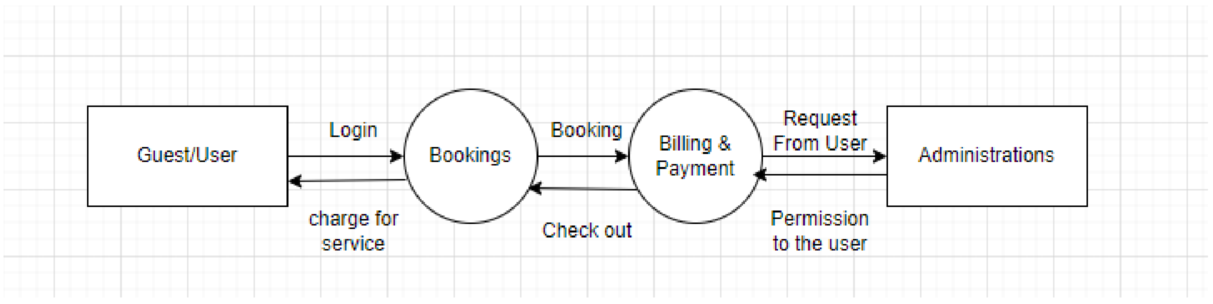
6. Plan the starting and ending dates for various activities.

Gantt chart:



7. Determine the critical path.

Critical Path:



System's capabilities to booking room and collect payment from user through the online net banking system and send this data to administration.

Rubric wise marks obtained:

Rubrics	1	2	3	4	5	Total
Marks	Complete implementation as asked	Complete implementation as asked Problem analysis	Complete implementation as asked Problem analysis Development of the Solution	Complete implementation as asked Problem analysis Development of the Solution Concept Clarity & understanding	Complete implementation as asked Problem analysis Development of the Solution Concept Clarity & understanding Correct answer to all questions	

Signature of Faculty:

Practical – 3

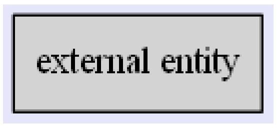
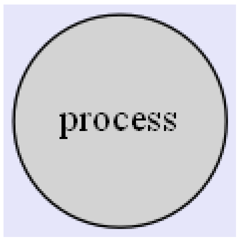
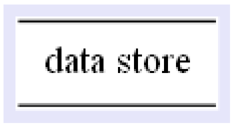
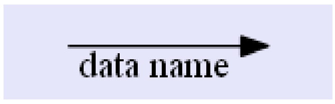
AIM: Prepare the following components of Data Flow Model: Data Dictionary, Data Flow Diagram and Structure Chart

Objectives:

1. To learn how to prepare data dictionary.
2. To learn how to draw data flow diagrams.
3. To convert a DFD into structure chart.

Theory:

- A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It can be manual, automated, or a combination of both.
- It shows how data enters and leaves the system, what changes the information, and where data is stored.
- The objective of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communication tool between a system analyst and any person who plays a part in the order that acts as a starting point for redesigning a system. The DFD is also called as a data flow graph or bubble chart.
- **The following observations about DFDs are essential:**
  - All names should be unique. This makes it easier to refer to elements in the DFD.
  - Remember that DFD is not a flow chart. Arrows is a flow chart that represents the order of events; arrows in DFD represents flowing data. A DFD does not involve any order of events.
  - Suppress logical decisions. If we ever have the urge to draw a diamond-shaped box in a DFD, suppress that urge! A diamond-shaped box is used in flow charts to represents decision points with multiple exists paths of which the only one is taken. This implies an ordering of events, which makes no sense in a DFD.
  - Do not become bogged down with details. Defer error conditions and error handling until the end of the analysis.
- Standard symbols for DFDs are derived from the electric circuit diagram analysis and are shown in fig:

Term	Notation	Remarks
External entity		Name of the external entity is written inside the rectangle
Process		Name of the process is written inside the circle
Data store		A left-right open rectangle is denoted as data store; name of the data store is written inside the shape
Data flow		Data flow is represented by a directed arc with its data name

Explanation of Symbols used in DFD

- **Process:** Processes are represented by circle. The name of the process is written into the circle. The name of the process is usually given in such a way that represents the functionality of the process. More detailed functionalities can be shown in the next Level if it is required. Usually it is better to keep the number of processes less than 7. If we see that the number of processes becomes more than 7 then we should combine some the processes to a single one to reduce the number of processes and further decompose it to the next level.
- **External entity:** External entities are only appear in context diagram. External entities are represented by a rectangle and the name of the external entity is written into the shape. These send data to be processed and again receive the processed data.
- **Data store:** Data stares are represented by a left-right open rectangle. Name of the data store is written in between two horizontal lines of the open rectangle. Data stores are used as repositories from which data can be flown in or flown out to or from a process.
- **Data flow:** Data flows are shown as a directed edge between two components of a Data Flow Diagram. Data can flow from external entity to process, data store to process, in between two processes and vice-versa.

.

- **Background / Preparation:**

Levels in Data Flow Diagrams (DFD)

The DFD may be used to perform a system or software at any level of abstraction. Infact, DFDs may be partitioned into levels that represent increasing information flow and functional detail. Levels in DFD are numbered 0, 1, 2 or beyond. Here, we will see primarily three levels in the data flow diagram, which are: 0-level DFD, 1-level DFD, and 2-level DFD.

0-level DFDM

It is also known as fundamental system model, or context diagram represents the entire software requirement as a single bubble with input and output data denoted by incoming and outgoing arrows. Then the system is decomposed and described as a DFD with multiple bubbles. Parts of the system represented by each of these bubbles are then decomposed and documented as more and more detailed DFDs. This process may be repeated at as many levels as necessary until the program at hand is well understood. It is essential to preserve the number of inputs and outputs between levels, this concept is called leveling by DeMacro. Thus, if bubble "A" has two inputs x_1 and x_2 and one output y, then the expanded DFD, that represents "A" should have exactly two external inputs and one external output as shown in fig:

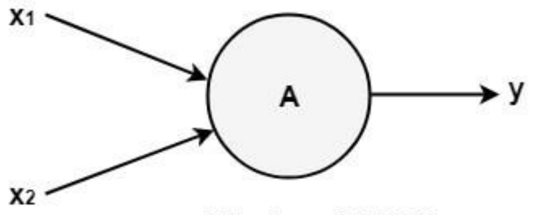


Fig: Level-0 DFD.

The Level-0 DFD, also called context diagram of the result management system is shown in fig. As the bubbles are decomposed into less and less abstract bubbles, the corresponding data flow may also be needed to be decomposed.

1-level DFD

In 1-level DFD, a context diagram is decomposed into multiple bubbles/processes. In this level, we highlight the main objectives of the system and breakdown the high-level process of 0-level DFD into subprocesses.

2-Level DFD

2-level DFD goes one process deeper into parts of 1-level DFD. It can be used to project or record the specific/necessary detail about the system's functioning.

- **Tools / Material Needed:**
    - **Hardware:**
    - **Software:**

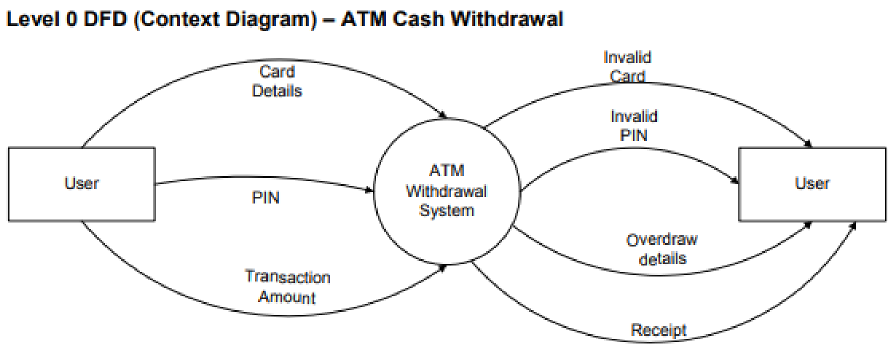
- **Procedure / Steps:**
    - **Example: ATM System**

The data flow diagram is a hierarchy of diagram consist of:

- Context Diagram (conceptually level zero)
- The Level-1 DFD
- And possible Level-2 DFD and further levels of functional decomposition depending on the complexity of your system.

Context DFD

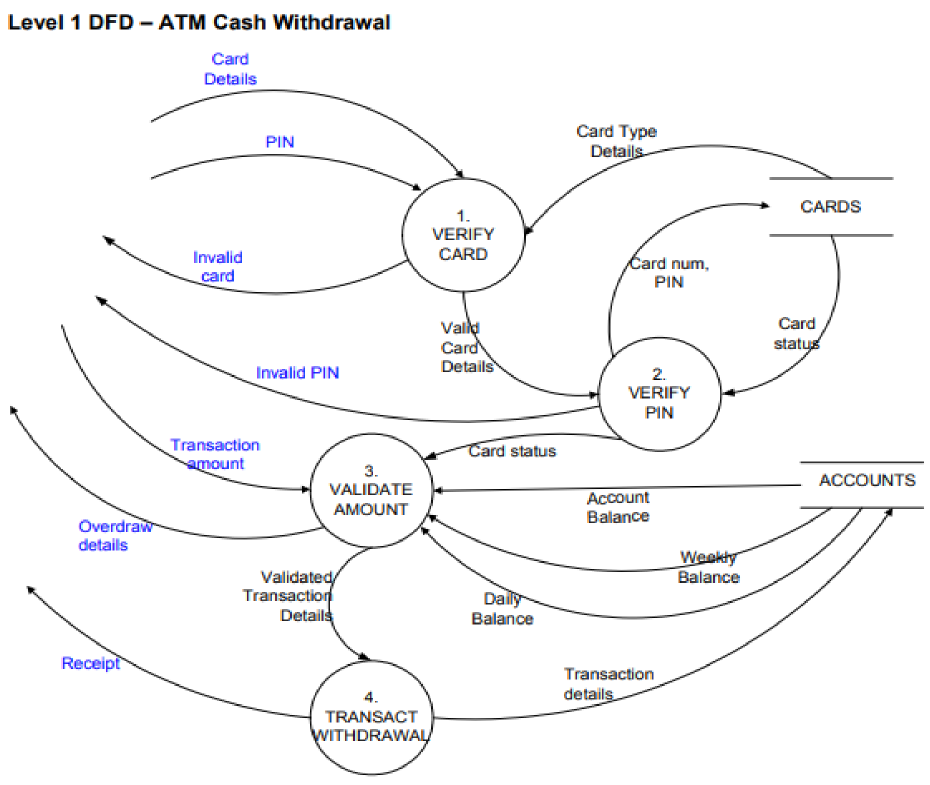
The figure below shows a context Data Flow Diagram that is drawn for an Android supermarket app. It contains a process (shape) that represents the system to model, in this case, the "*ATM System*". It also shows the participants who will interact with the system, called the external entities. In this example, there is only one external entity, which is the *User*. In between the process and the external entity, there is a uni-directional connector, which indicates the existence of information exchange between user and the ATM System, and the information flow is bi-directional.



Context DFD is the entrance of a data flow model. It contains one and only one process and does not show any data store, which makes the diagram simple.

Level 1 DFD

The figure below shows the level 1 DFD, which is the decomposition (i.e. break down) of the ATM System process that is shown in the context DFD. Read through the diagram and then we will introduce some of the key concepts based on this diagram.



Quiz:

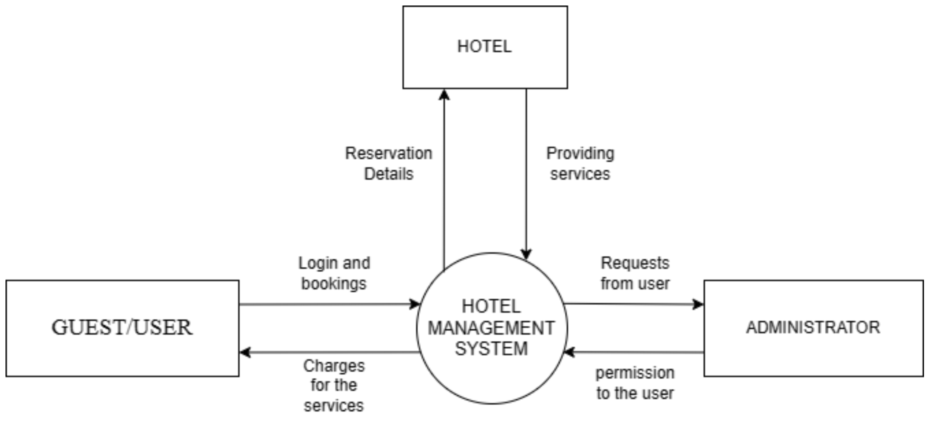
1) Draw the Data Flow Diagram for Hotel Management System.
2) What is DFD? Give advantage and disadvantages of DFD.
3) Types and Components of Data Flow Diagram (DFD)

Suggested Reference:

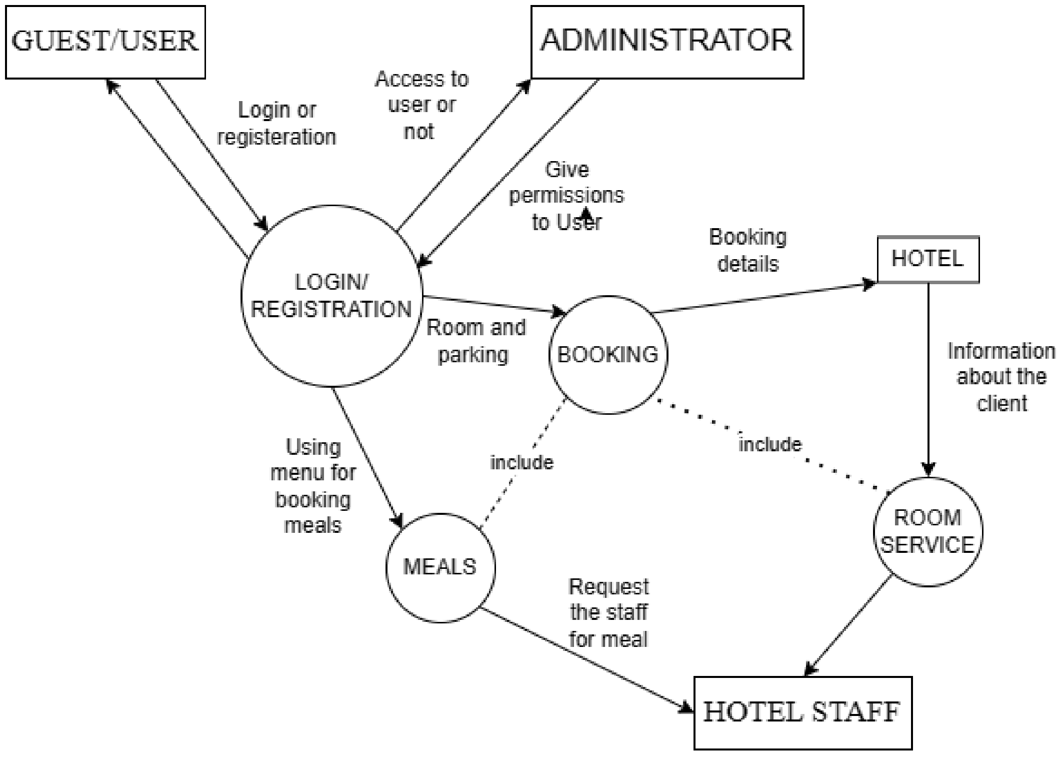
1) NPTEL course material - System Analysis and Design
2) Booch, G. et al. The Unified Modeling Language User Guide. Chapters 15, 18, 27. Addison-Wesley.
3) Jacobson, I. et al. Object-Oriented Software Engineering: A Use-Case Driven Approach. Addison-Wesley.
4) Fowler, M. UML Distilled: A Brief Guide to the Standard Object Modelling Language. Chapter 5. Addison Wesley.

Data Flow Diagram (DFD)

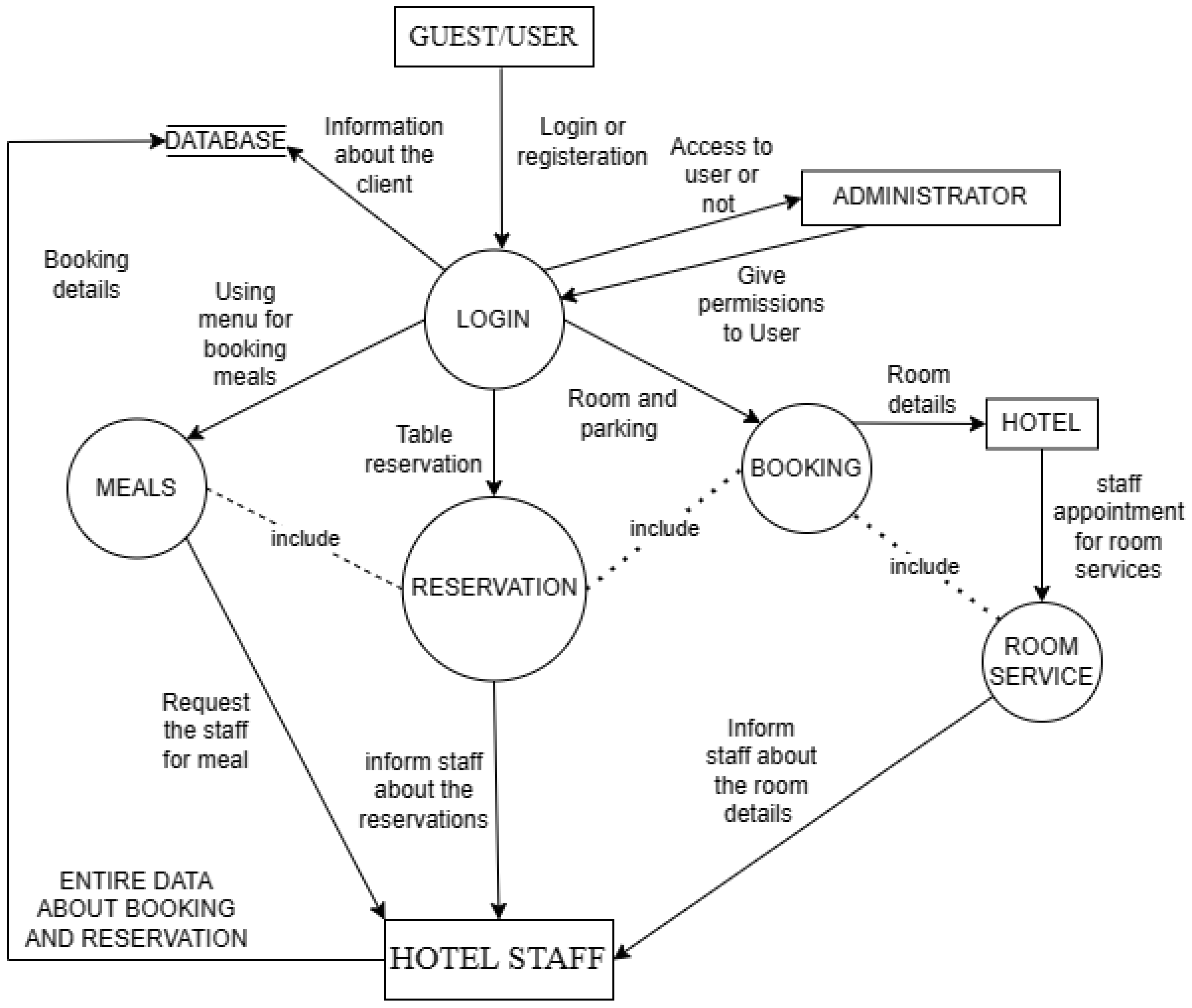
- Level 0 :-



- Level 1:-



- Level 2 :-


GUEST/USER
Login or registeration
DATABASE
Information about the client
Access to user or not
ADMINISTRATOR
Give permissions to User
LOGIN
Booking details
Using menu for booking meals
MEALS
Table reservation
RESERVATION
Room and parking
BOOKING
Room details
HOTEL
staff appointment for room services
include
include
include
ROOM SERVICE
Request the staff for meal
inform staff about the reservations
Inform staff about the room details
ENTIRE DATA ABOUT BOOKING AND RESERVATION
HOTEL STAFF

Data Dictionary

User_Database					
Sr no.	Name	Description	DataType	Length	Validation rules
1	id	User unique id	integer	128	Not Null, Auto Generated
2	Name	user name	varchar	150	Not Null
3	Contact_no	Mobile no	integer	12	Not Null
4	Email	User Email Address	varchar	256	Not Null
5	password	User Password	varchar	150	Not Null, Encreptyed Form

User_History Database					
Sr no.	Name	Description	DataType	Length	Validation rules
1	id	Database unique id	integer	128	Not Null, Auto Generated
2	User_id	User unique id	integer	128	Not Null
3	Date_Time	Time Duration of Customer	varchar	150	Not Null
4	Bill_Id	Payment details	integer	128	Not Null, Auto Fill
5	Member	No of member	integer	50	Not Null

User_Payment Database					
Sr no.	Name	Description	DataType	Length	Validation rules
1	id	Database unique id	integer	128	Not Null, Auto Generated
2	User_id	User unique id	integer	128	Not Null
3	Date_Time	Time Duration of Customer	varchar	150	Not Null
4	Amount	Payable amount	float	50	Not Null

Rubric wise marks obtained:

Rubrics	1	2	3	4	5	Total
Marks	Complete implementation as asked	Complete implementation as asked Problem analysis	Complete implementation as asked Problem analysis Development of the Solution	Complete implementation as asked Problem analysis Development of the Solution Concept Clarity & understanding	Complete implementation as asked Problem analysis Development of the Solution Concept Clarity & understanding Correct answer to all questions	

Signature

Practical – 4

AIM: Prepare the user's view analysis : Describe different scenarios and Draw Use case diagrams using UML

- **Objectives:**
    1. To write different scenarios of the system's execution.
    2. To explore various UML use case diagram components to draw USECASE diagram.

- **Theory:**
    - A use case diagram is used to represent the dynamic behavior of a system. It encapsulates the system's functionality by incorporating use cases, actors, and their relationships. It models the tasks, services, and functions required by a system/subsystem of an application. It depicts the high-level functionality of a system and also tells how the user handles a system.
    - Purpose of Use Case Diagrams
        - The main purpose of a use case diagram is to portray the dynamic aspect of a system. It accumulates the system's requirement, which includes both internal as well as external influences. It invokes persons, use cases, and several things that invoke the actors and elements accountable for the implementation of use case diagrams. It represents how an entity from the external environment can interact with a part of the system.

    Following are the purposes of a use case diagram given below:
        1. It gathers the system's needs.
        2. It depicts the external view of the system.
        3. It recognizes the internal as well as external factors that influence the system.
        4. It represents the interaction between the actors.
    - In a use-case diagram, an **actor** is a user of the system (i.e. Something external to the system; can be human or non-human) acting in a particular role.
    - A **use-case** is a task which the **actor** needs to perform with the help of the system, e.g., find details of a book or print a copy of a receipt in a bookshop.
    - We can draw a box (with a label) around a set of use cases to denote the system boundary, as on the previous slide ("library system").

    Inheritance can be used between actors to show that all use cases of one actor are available to the other:
    If several use cases include, as part of their functionality, another use case, we have a special way to show this in a use-case diagram with an <<**include**>> relation.
    If a use-case has two or more significantly different outcomes, we can show this by **extending** the use case to a main use case and one or more subsidiary cases.

- **Background / Preparation:**

How to draw a Use Case diagram?

It is essential to analyze the whole system before starting with drawing a use case diagram, and then the system's functionalities are found. And once every single functionality is identified, they are then transformed into the use cases to be used in the use case diagram.

After that, we will enlist the actors that will interact with the system. The actors are the person or a thing that invokes the functionality of a system. It may be a system or a private entity, such that it requires an entity to be pertinent to the functionalities of the system to which it is going to interact.

Once both the actors and use cases are enlisted, the relation between the actor and use case/ system is inspected. It identifies the no of times an actor communicates with the system. Basically, an actor can interact multiple times with a use case or system at a particular instance of time.

Following are some rules that must be followed while drawing a use case diagram:

1. A pertinent and meaningful name should be assigned to the actor or a use case of a system.
2. The communication of an actor with a use case must be defined in an understandable way.
3. Specified notations to be used as and when required.
4. The most significant interactions should be represented among the multiple no of interactions between the use case and actors.

The purposes of use case diagrams can be as follows:

- Used to gather requirements of a system.
- Used to get an outside view of a system.
- Identify external and internal factors influencing the system.
- Show the interacting among the requirements are actors.

Scenarios

- **Scenarios** are real-life examples of how a system can be used.
- They should include
  - A description of the starting situation;
  - A description of the normal flow of events;
  - A description of what can go wrong;
  - Information about other concurrent activities;

A description of the state when the scenario finishes.

- **Tools / Material Needed:**
  - **Hardware:**
  - **Software:**

- **Procedure / Steps:**
  - **Developing Use Cases:**
  - Step One – Define the set of actors that will be involved in the story

        - Actors are people, devices, or other systems that use the system or product within the context of the function and behavior that is to be described
        - Actors are anything that communicate with the system or product and that are external to the system itself
    - Step Two – Develop use cases, where each one answers a set of questions

        - Example: ATM System

            1. Identification of use cases.
            2. Identification of actors.
            3. Sample input:
                1. Actor: User
                2. Use case: Login

- Withdrawal Use Case :

  A withdrawal transaction asks the customer to choose a type of account to withdraw from (e.g. checking) from a menu of possible accounts, and to choose a dollar amount from a menu of possible amounts. The system verifies that it has sufficient money on hand to satisfy the request before sending the transaction to the bank. (If not, the customer is informed and asked to enter a different amount.) If the transaction is approved by the bank, the appropriate amount of cash is dispensed by the machine before it issues a receipt. A withdrawal transaction can be cancelled by the customer pressing the Cancel key any time prior to choosing the dollar amount.
- Deposit Use Case :

  A deposit transaction asks the customer to choose a type of account to deposit to (e.g. checking) from a menu of possible accounts, and to type in a dollar amount on the keyboard. The transaction is initially sent to the bank to verify that the ATM can accept a deposit from this customer to this account. If the transaction is approved, the machine accepts an envelope from the customer containing cash and/or checks before it issues a receipt. Once the envelope has been received, a second message is sent to the bank, to confirm that the bank can credit the customer‟ account – contingent on manual verification of the deposit envelope contents by an operator later.

  A deposit transaction can be cancelled by the customer pressing the Cancel key any time prior to inserting the envelope containing the deposit. The transaction is automatically cancelled if the customer fails to insert the envelope containing the deposit within a reasonable period of time after being asked to do so.

- Transfer UseCase:

  A transfer transaction asks the customer to choose a type of account to transfer from (e.g. checking) from a menu of possible accounts, to choose a different account to transfer to, and to type in a dollar amount on the keyboard. No further action is required once the transaction is approved by the bank before printing the receipt.

  A transfer transaction can be cancelled by the customer pressing the Cancel key any time prior to entering a dollar amount.

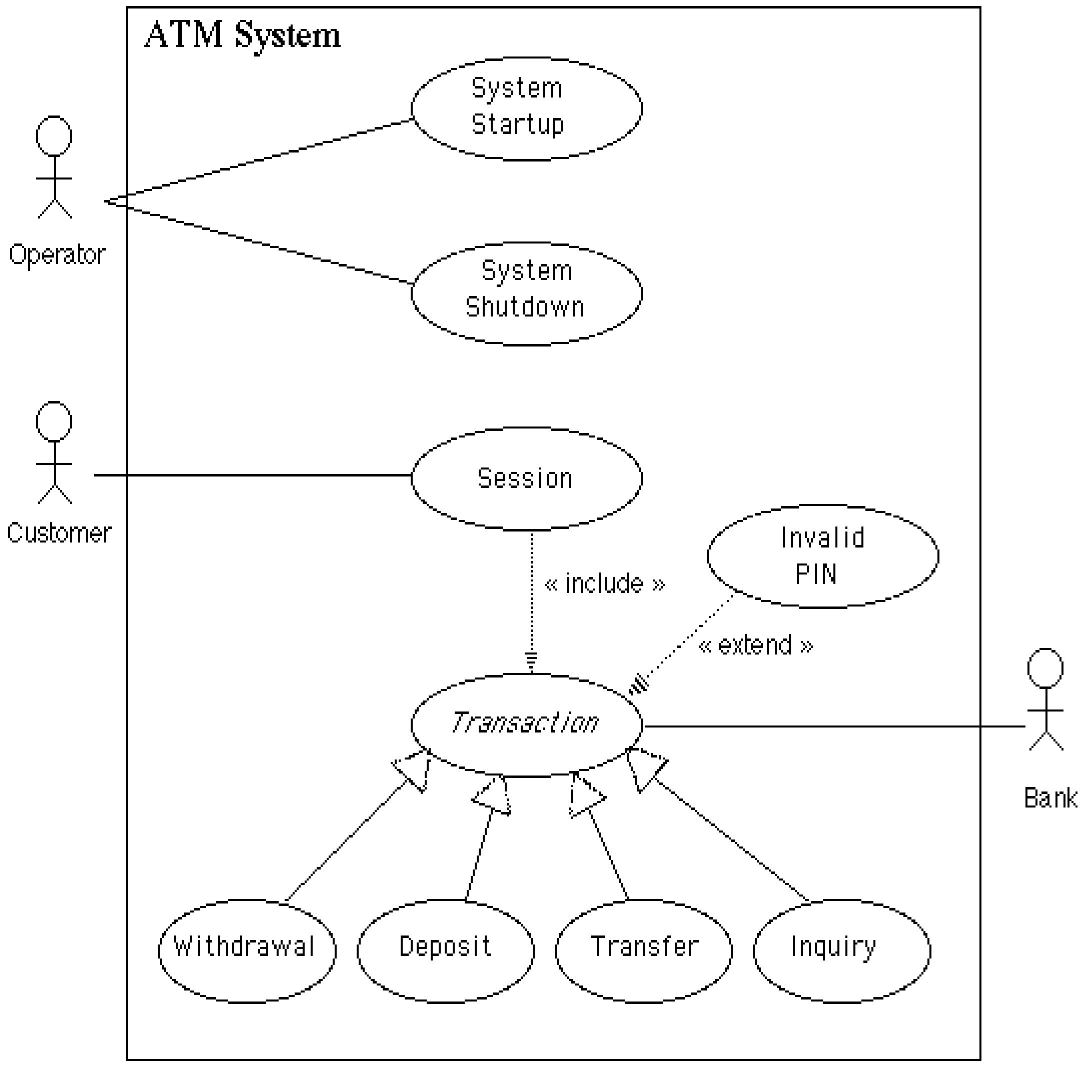
- Inquiry Use Case :

  An inquiry transaction asks the customer to choose a type of account to inquire about from a menu of possible accounts. No further action is required once the transaction is approved by the bank before printing the receipt. An inquiry transaction can be cancelled by the customer pressing the Cancel key any time prior to choosing the account to inquire about.

- Validate User usecase:

  This usecase is for validate the user i.e check the pin number, when the bank reports that the customer‟s transaction is disapproved due to an invalid PIN. The customer is required to re-enter the PIN and the original request is sent to the bank again. If the bank now approves the transaction, or disapproves it for some other reason, the original use case is continued; otherwise the process of re-entering the PIN is repeated. Once the PIN is successfully re-entered.

  If the customer fails three times to enter the correct PIN, the card is permanently retained, a screen is displayed informing the customer of this and suggesting he/she contact the bank, and the entire customer session is aborted.



Quiz:

1. What is use case diagram? Draw use case for library management system.
2. List relationship used in use case.
3. What Tests Can Help Find Useful Use Cases?

Suggested Reference:

1) Use Case Diagrams.
2) Unified Modeling Language, Superstructure, V2.1.2.
3) "Functional Requirements and Use Cases", Ruth Malan and Dana Bredemeyer, Bredemeyer Consulting.

4) "A Use Case Template: draft for discussion", Derek Coleman, Hewlett-Packard Software Initiative X. J. Zheng, X. Liu, & S. Liu. (2010). Use Case and Non-functional Scenario Template-Based Approach to Identify Aspects. Computer Engineering and Applications ICCEA 2010 Second International Conference on (Vol. 2, pp. 89-93).

Describe different scenarios:

Scenarios in which our system or application interacts with people, organizations, or external systems. Goals that our system or application helps those entities achieve.

The scope of our System:

1. **Guest Check-In:**
   - Scenario: A guest arrives at the hotel and wants to check-in.
   - Use Case: The system allows the guest to provide identification, verify their reservation, assign a room, and issue a room key or access details.
2. **Guest Check-Out:**
   - Scenario: A guest is ready to check out.
   - Use Case: The system calculates the final bill, processes payment, and checks the guest out while making the room available for cleaning and reassignment.
3. **Room Reservation:**
   - Scenario: A guest plans to stay at the hotel and wants to reserve a room.
   - Use Case: The system provides the guest with options to select dates, room types, and preferences, and confirms the reservation.
4. **Room Service Request:**
   - Scenario: A guest staying in the room needs room service.
   - Use Case: The system allows the guest to request services such as food delivery, housekeeping, or maintenance and provides updates on the request status.
5. **Billing and Payment:**
   - Scenario: A guest has completed their stay and needs to settle the bill.
   - Use Case: The system calculates the bill, applies discounts or promotions, processes payments, and provides a receipt.
6. **Admin Functions:**
   - Scenario: Hotel staff needs to manage various administrative tasks.
   - Use Case: The system enables staff to manage reservations, allocate rooms, manage employee accounts, configure system settings, and generate reports.
7. **Multiple Properties:**
   - Scenario: A hotel chain operates multiple properties.
   - Use Case: The system allows central management of multiple properties, enabling staff to oversee and coordinate operations across different locations.
8. **Group Bookings:**
   - Scenario: A group or event organizer wants to make multiple room reservations.

- Use Case: The system facilitates group bookings, allowing for the reservation of multiple rooms and coordination of group-related services.

9. **Customer Relationship Management (CRM):**

- Scenario: The hotel wants to maintain good relationships with returning guests.
- Use Case: The system stores guest preferences, booking history, and special requests, enabling personalized service and offers for loyal customers.

10. **Online Booking Portal:**

- Scenario: Potential guests want to make reservations online.
- Use Case: The system provides an online booking portal accessible through the hotel's website or mobile app, allowing guests to book rooms and services.

11. **Guest Feedback and Reviews:**

- Scenario: Guests want to provide feedback or leave reviews.
- Use Case: The system allows guests to submit feedback and reviews, which can be used for quality improvement and reputation management.

12. **Maintenance and Repairs:**

- Scenario: A room needs maintenance or repairs.
- Use Case: The system enables staff to log and track maintenance requests, assign them to maintenance personnel, and monitor their completion.

Draw Use case diagrams using UML:

- Use case diagram model the behavior of a system. Used ditto illustrate the fictional requirements of the system and its interaction with external agents(actors).
- A use case diagram gives us a high-level view of the system without using into implementation details. The Purpose of a use Case diagram in UML is to demonstrate the different ways that a user might interact with a system.

Procedure / Steps:

Step One – Define the set of actors that will be involved in the story

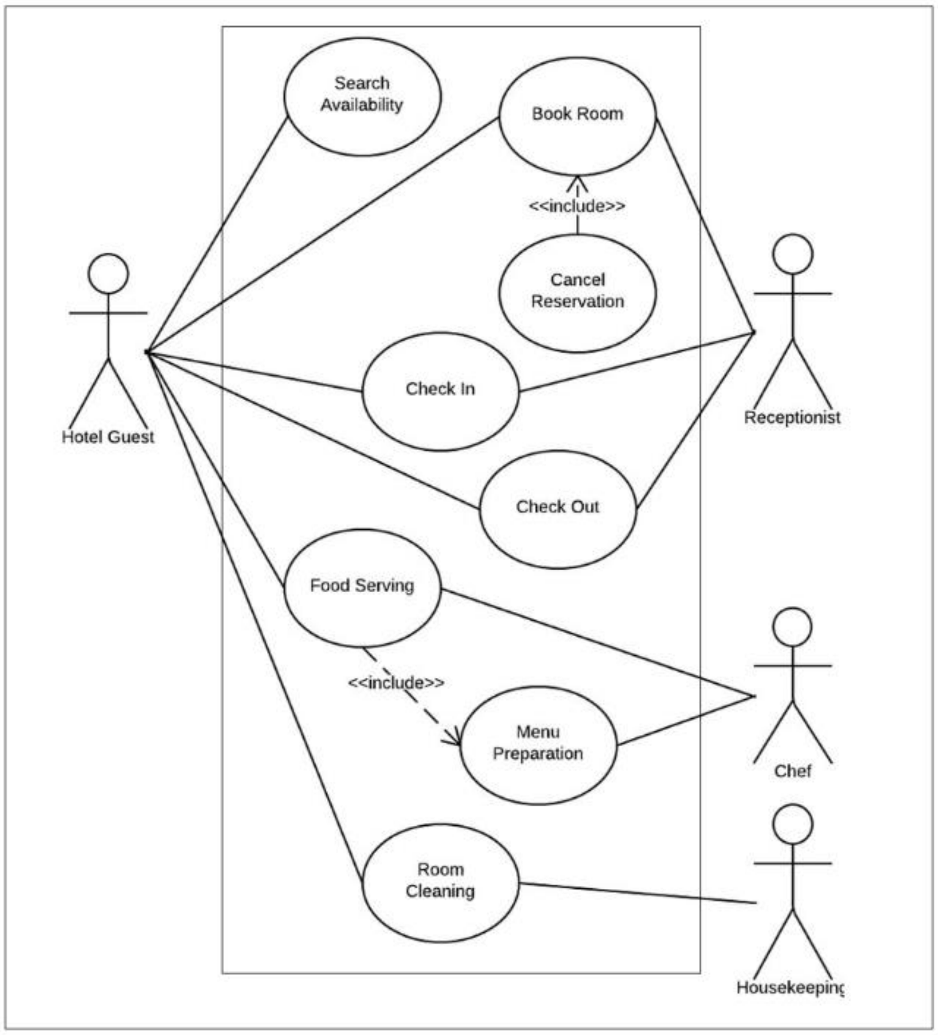
- **Actors:** Visitor, Admin, User
- **Patient**: A person who seeks Hotel’s services.
- **Hotel staff/Service Provider**: Staff providing Hotel services.
- **Administrator**: Personnel responsible for system administration and configuration.

Step Two – Develop use cases, where each one answers a set of questions

Use Cases:

1. **User login:**
   - Actor: Guest, Administrator
   - Description: Our Guest will enter the username assign to him/her and the verify it using password and then out administrator will check it in the system and then allow access to the guest for further use of the software.
2. **Registration:**
   - Actor: Guest, Administrator
   - Description: Guest will register themselves to the system and then administrator will register the entry and save it to the system, so guest can get a login.
3. **Online Booking:**
   - Actor: Guest
   - Description: System will redirect then to the booking option where guest can book rooms and parking space according to their arrival.
4. **Reservations:**
   - Actor: Guest, Staff
   - Description: Guest will reserve the table and staff for the comfortable experience.
5. **Meals Management:**
   - Actor: Guest, Staff
   - Description: Guest will order for meal before arrival to the dinning area and can also specify the chief assign to their meal.
6. **Check-in process**
   - Actor: Guest, Staff
   - Description: After booking room in the hotel guest are suppose to check-in the rooms so that staff can understand guest is in the room or not for cleaning purpose.
7. **Feedback and review**
   - Actor: Guest
   - Description: Guest will share their experience on social media platform of the hotel and also review hotel services on online portal.

Use Cases Diagram:



Rubric wise marks obtained:

Rubrics	1	2	3	4	5	Total
Marks	Complete implementation as asked	Complete implementation as asked Problem analysis	Complete implementation as asked Problem analysis Development of the Solution	Complete implementation as asked Problem analysis Development of the Solution Concept Clarity & understanding	Complete implementation as asked Problem analysis Development of the Solution Concept Clarity & understanding Correct answer to all questions	

Signature