

Department of Computer

Engineering Academic Term: First

Term 2023-24

Class: T.E /Computer Sem – V / Software Engineering

Practical No:	2
Title:	Data Flow Analysis of the Project in Software Engineering
Date of Performance:	17-08-23
Roll No:	9595
Team Members:	Atharva Dalvi

Rubrics for Evaluation:

Sr. No	Performance Indicator	Excellent	Good	Below Average	Total Score
1	On time Completion & Submission (01)	01 (On Time)	NA	00 (Not on Time)	
2	Theory Understanding(02)	02(Correct)	NA	01 (Tried)	
3	Content Quality (03)	03(All used)	02 (Partial)	01 (rarely followed)	
4	Post Lab Questions (04)	04(done well)	3 (Partially Correct)	2(submitted)	

Signature of the Teacher

Lab Experiment 06

Experiment Name:

Data Flow Analysis of the Project in Software Engineering

Objective:

The objective of this lab experiment is to introduce students to Data Flow Analysis, a technique used in software engineering to understand the flow of data within a software system. Students will gain practical experience in analyzing the data flow of a sample software project, identifying data dependencies, and modeling data flow diagrams.

Introduction:

Data Flow Analysis is a vital activity in software development, helping engineers comprehend how data moves through a system, aiding in identifying potential vulnerabilities and ensuring data integrity.

Lab Experiment Overview:

1. Introduction to Data Flow Analysis: The lab session begins with an overview of Data Flow Analysis, its importance in software engineering, and its applications in ensuring data security and accuracy.
2. Defining the Sample Project: Students are provided with a sample software project, which includes the data elements, data stores, processes, and data flows.
3. Data Flow Diagrams: Students learn how to construct Data Flow Diagrams (DFDs) to visualize the data flow in the software system. They understand the symbols used in DFDs, such as circles for processes, arrows for data flows, and rectangles for data stores.
4. Identifying Data Dependencies: Students analyze the sample project and identify the data dependencies between various components. They determine how data is generated, processed, and stored in the system.
5. Constructing Data Flow Diagrams: Using the information gathered, students create Data Flow Diagrams that represent the data flow within the software system. They include both high-level context diagrams and detailed level-0 and level-1 diagrams.
6. Data Flow Analysis: Students analyze the constructed DFDs to identify potential bottlenecks, inefficiencies, and security vulnerabilities related to data flow.
7. Conclusion and Reflection: Students discuss the significance of Data Flow Analysis in software development and reflect on their experience in constructing and analyzing Data Flow Diagrams.

Learning Outcomes:

By the end of this lab experiment, students are expected to:

Understand the concept of Data Flow Analysis and its importance in software engineering.

Gain practical experience in constructing Data Flow Diagrams to represent data flow in a software system.

Learn to identify data dependencies and relationships within the software components.

Develop analytical skills to analyze Data Flow Diagrams for potential issues and vulnerabilities.

Appreciate the role of Data Flow Analysis in ensuring data integrity, security, and efficiency.

Pre-Lab Preparations:

Before the lab session, students should familiarize themselves with Data

Flow Analysis concepts and the symbols used in Data Flow Diagrams. They should review data dependencies and data flow modeling in software systems.

Materials and Resources:

Project brief and details for the sample software project

Whiteboard or projector for constructing Data Flow Diagrams

Drawing tools or software for creating the diagrams

Conclusion: The lab experiment on Data Flow Analysis of a software project equips students with essential skills in understanding data flow within a system. By constructing and analyzing Data Flow Diagrams, students gain insights into how data is processed, stored, and exchanged, enabling them to identify potential issues and security concerns. The practical experience in Data Flow Analysis enhances their ability to design efficient and secure software systems that ensure data integrity and meet user requirements. The lab experiment encourages students to apply Data Flow Analysis in real- world software development projects, promoting better data management and system design practices.

Customers and hotel staff interact with the system, representing the system's external interfaces.

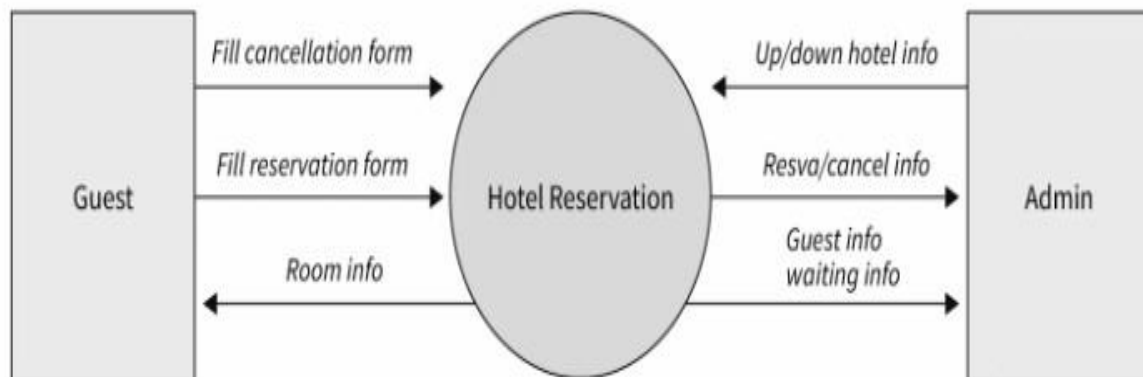
The processes represent the core functionality of the reservation system. They ensure that reservation requests are handled, availability is checked, reservations are confirmed or canceled, and guests are checked in or out.

The reservation database stores essential information about reservations and guests, while the room inventory database keeps track of room availability.

Data flows between processes and data stores, carrying information required for reservation handling.

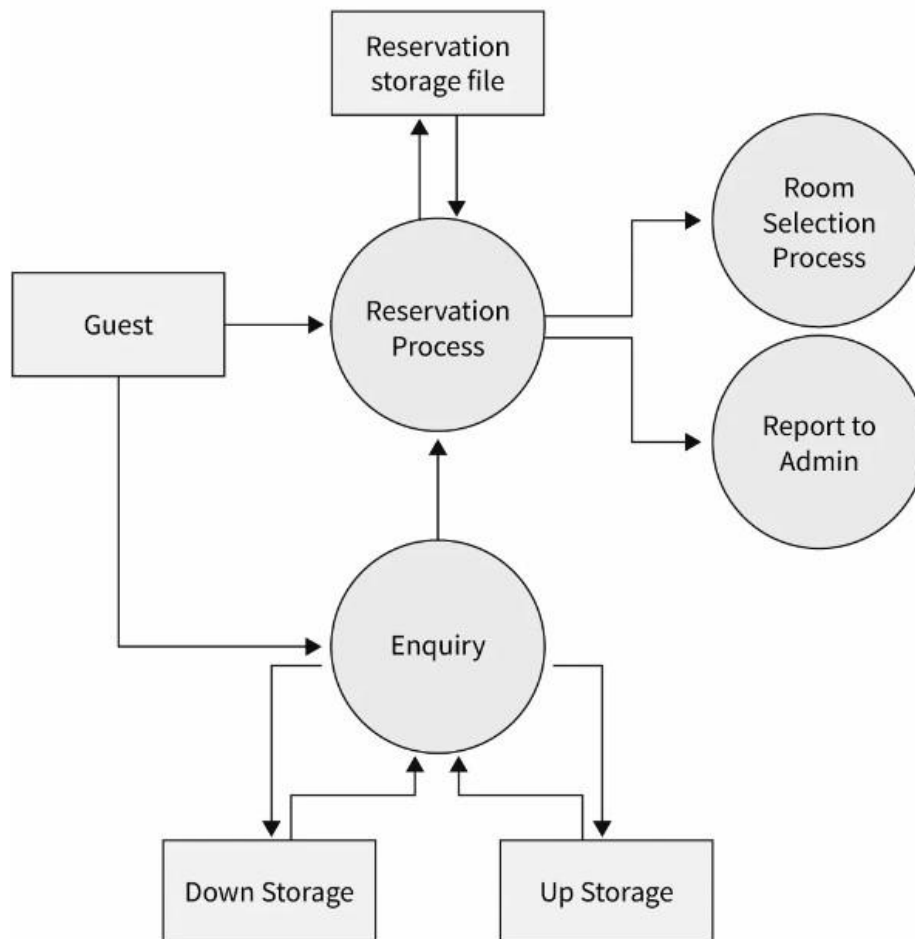
Level 0 DFD

It's also referred to as a context diagram. It is intended to be an abstract view, depicting the system as a single process with external elements. It represents the complete system as a single bubble with incoming and outgoing arrows indicating input and output data.



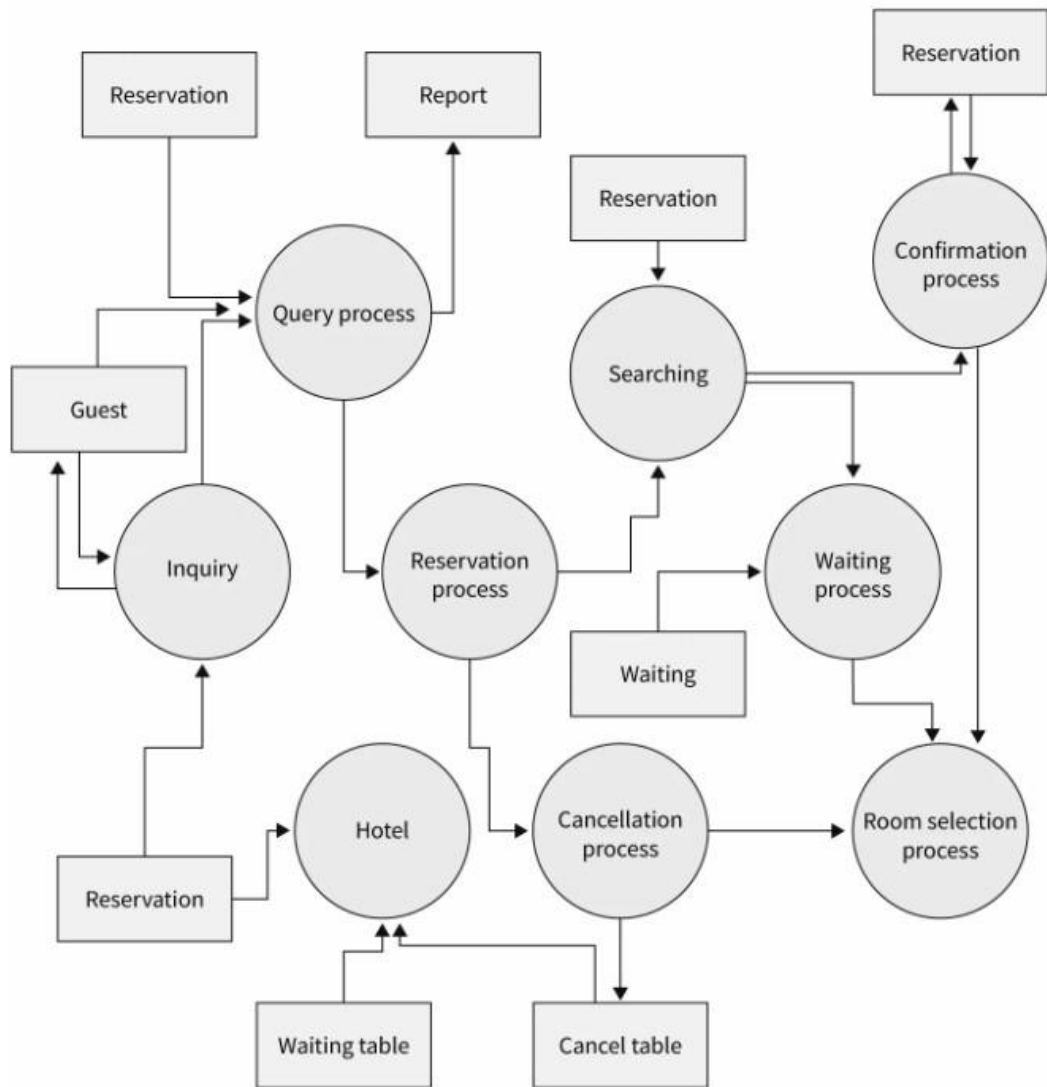
Level 1 DFD


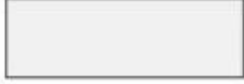






In 1-level DFD, the context diagram is broken down into many bubbles and processes. At this level, we highlight the system's essential functions and divide the high-level process of 0-level DFD into subprocesses.



Level 2 DFD

2-level DFD delves deeper into aspects of 1-level DFD. It can be used to design or record specific/necessary details about how the system works



Notation	De Marco & Yourdon	Gane and Sarson
External Entity		
Process		
Data Store		
Data Flow		

a) Evaluate the benefits of using Data Flow Diagrams (DFD) to analyze and visualize the data movement in a complex software system.

- Clarity and Simplicity:**
 DFDs provide a clear and simplified visual representation of how data flows within a system. They use straightforward symbols and arrows to show how data moves between processes, data stores, and external entities, making complex systems easier to understand.
- Communication:**
 DFDs serve as a common language for communication between various stakeholders, including developers, analysts, and end-users. They facilitate discussions about system requirements and behavior.
- Requirements Analysis:**
 DFDs are instrumental in gathering and analyzing requirements. They help identify input sources, output destinations, and data transformations required for system functionality.

- **Identifying Data Sources and Sinks:**
DFDs help pinpoint where data originates (sources) and where it is consumed or stored (sinks). This is crucial for understanding data dependencies.
- **Data Security:**
DFDs can highlight potential data security vulnerabilities by showing how sensitive data flows within the system. This allows for early identification of security risks.

b) Apply data flow analysis techniques to a given project and identify potential data bottlenecks and security vulnerabilities.

Construct DFD:

Create a comprehensive Data Flow Diagram (DFD) for the given project, including external entities, processes, data stores, and data flows.

1. Data Flow Analysis:

Analyze the DFD to identify potential bottlenecks:

Look for processes that handle a disproportionately high volume of data.

Identify areas where data flows converge, potentially causing congestion.

Pay attention to data stores that are frequently accessed or updated.

2. Security Analysis:

Identify potential security vulnerabilities:

Trace the flow of sensitive data within the system to see where it's processed, stored, and transmitted.

Identify points where unauthorized access or data leakage could occur.

Check if encryption and access control measures are in place for sensitive data.

c) Propose improvements to the data flow architecture to enhance the system's efficiency and reduce potential risks.

To enhance system efficiency and reduce potential risks:

Optimize Processes: If bottlenecks are identified, consider optimizing the processes responsible for high data volumes. This might involve improving algorithms, utilizing caching mechanisms, or distributing processing tasks.

Parallel Processing: Implement parallel processing for tasks that can benefit from it, reducing processing times and potential bottlenecks.

Data Compression: Implement data compression techniques to reduce the size of data being transferred between processes or stored in data stores.

Load Balancing: Use load balancing strategies to evenly distribute data processing tasks across multiple servers or resources.

Enhance Security Measures: Strengthen data security by implementing encryption, access controls, and auditing mechanisms at vulnerable points in the data flow.

Data Minimization: Minimize the storage of sensitive data and ensure data retention policies comply with privacy regulations to reduce security risks.

Regular Audits: Conduct regular security audits and performance assessments to identify and address potential issues as the system evolves.

Documentation: Keep the DFD updated as the system changes and evolves. It serves as a valuable reference for understanding data flow and system architecture.