**HOSTEL ADMINISTRATION SYSTEM**

**Project work submitted in partial fulfillment of the**

**requirements for the degree of**

**Bachelor of Science in Computer Science**

**to**

**Thiruvalluvar University, Serkkadu, Vellore-632 115**

**Submitted by**

**40720U18014 – EZHILAN E**

**40720U18015 - EZHUMALAI J**

**40720U18026 – MANISHKUMAR V**

**Under the Guidance of**

**Mrs. D. SASIKALA M.Sc., M.Phil., M.Tech., SET.,**

**Assistant Professor, Department of Computer Science**



**APRIL / MAY - 2023**

**SRIMATH SIVAGNANA BALAYA SWAMIGAL TAMIL, ARTS, SCIENCE COLLEGE, Mailam-604304.**

**(AFFILIATED TO THIRUVALLUVAR UNIVERSITY)**

**Serkkadu, Vellore-632 115**

**BONAFIDE CERTIFICATE**

This is to certified that the report entitled **“HOSTEL ADMINISTRATION SYSTEM”** being submitted to the Thiruvalluvar University, Vellore by **40720U18014 EZHILAN E, 40720U18015 EZHUMALAI J** and **40720U18026 MANISHKUMAR V** for the fulfilment for the award of degree of **B.Sc., COMPUTER SCIENCE** is a bonafide record work carried out under my direct supervision and guidance during the year 2022 - 2023 .

PROJECT GUIDE HEAD OF THE DEPARTMENT

Submitted for the Viva-Voce examination held on\_\_\_\_\_\_\_\_\_\_\_\_\_\_

INTERNAL EXAMINER EXTERNAL EXAMINER

**CHAPTER 2**

**SYSTEM ANALYSIS**

# 2.1 EXISTING SYSTEM

The existing system of Hostel Management system involves lots and lots of paper work. In the existing system the book maintained is maintain in manually. The student details, visitor details, outings details, fess renewal details, mess employee details, warden details are maintained in manually. The study of the existing system revealed that the system has several drawbacks.

# The limitations in Existing system are:

* The records stored manually can be altered by unauthorized person
* Problem arises if any of the records get misplaced.
* The searching time is waste.
* Reservation of room is not possible.
* Register is maintained to trace the students accessing the hostel.
* Overload of staff members.

# NEED FOR COMPUTERIZED SYSTEM:

**The need of computerization arises due to the following reasons:**

* The required information can be retrieved easily.
* The details can be stored in the storage permanently.
* Correction can be made easily.
* High speed.
* Wastage of manpower is reduce.
* Large volume of records to process.
* Less time.

# 2.2 PROPOSED SYSTEM

The proposed system gives the details about the students, wardens, and the mess employee in hostel. This system contains four main modules such as details maintenance, visitors, fees renewal and outings.

The details maintenance module is used to maintain the details of the students. The Existing student module is used to search the availability of the students. The students can be searched based on student ID. The visitors module is used to visitors, visit the student based on its student IDs. The outings module is used to notice the time of in and out time of the student based on its student .The fees renewal module is used to renewal the fees of the student. The mess employee module is used to the mess employee details. The warden module is used to maintain the warden details in the student hostel.

# FEASIBILITY STUDY:

It is a test of a proposed system according to its workability, impact on the organization.

It is a test whether the system meet user needs and using the resources properly.

Depending on the results of initial investigation, We came to know that the Hostel Management wants to manage large amount of records accurately, precisely and quickly with security and integrity. The administration wants a system that can provide for the management of the currently active records anywhere and anytime. To accomplish this task, we need a Relational Database Management System (RDBMS) and a Web based GUI application development environment. We did three types of  
 feasibility study:

* Technical feasibility
* Operational feasibility
* Economic feasibility

# Technical feasibility:

In this feasibility we have to find out weather all the technical resources are available or not for the development of the system. In the College Campus all types of technical facility were available. All the software and hardware were available for the development viewpoint. There are a number of technical issues which are generally raised during the feasibility stage of the investigation. They are as follows:

* Does the necessary technology exist to do what is suggested and can it be acquired.
* Does the proposed equipment have the technical capacity to hold the data required to use the new system?
* Can the system be upgraded if developed?
* Are there technical guarantees of accuracy, reliability, ease of access and data security

**Operational feasibility:**

Here we have to find out whether our system is fulfilling the user requirements or not. For this, discussions were done with the person incharge of hostel. Proposed projects are beneficial only if they can be turned into information systems that will meet the operating requirements of the organization. This test of feasibility asks if the system will work when it is developed and installed. Some of the important questions that are useful to test the operational feasibility of a project are given below:

• Is there sufficient support for the project from the management?

* Are current business methods acceptable to the users?

* Have the users been involved in the planning and development?

* Will the proposed system cause harm? Will it produce poorer result in any case or area?

# Economic feasibility:

It is seen whether the expenditure incurred for developing the new system will be cost effective or not.

This basically involves the top level management of the company who are the decision maker.

Following decision is made during Economical feasibility:

• The cost to conduct a full system investigation.

• The cost of hardware and software for the class of application being considered.

**CHAPTER 3**

**SYSTEM DESIGN AND DEVELOPMENT**

System design and development involves creating a software solution to solve a problem. The documentation provides a detailed account of the process and serves as a reference guide for stakeholders. It includes information about the system's purpose, architecture, technologies used, development process, and instructions on how to use the system. Documentation is crucial for ensuring a well-designed, implemented, and tested system and provides a reference for future maintenance and improvements.

**FILE DESIGN**

File design refers to the process of creating a logical and efficient structure for storing and organizing data within a computer system. The goal of file design is to ensure that data is stored in a way that is easy to access, retrieve, and maintain. This process involves determining the type of data that needs to be stored, the format in which it should be stored, and the relationships between different pieces of data.

**Simplicity:** The file design should be easy to understand and maintain, with a minimal number of data structures and relationships.

**Efficiency:** The file design should be optimized for fast access and retrieval of data.

**Flexibility:** The file design should be flexible enough to accommodate changes in the data structure or the system requirements.

**Security:** The file design should ensure that data is protected from unauthorized access or modification.

**Scalability:** The file design should be able to handle large amounts of data as the system grows and evolve.

**Database Design:** SQLite is a database engine. It is software that allows users to interact with a relational database. In SQLite, a database is stored in a single file — a trait that distinguishes it from other database engines. This fact allows for a great deal of accessibility: copying a database is no more complicated than copying the file that stores the data, sharing a database can mean sending an email attachment.

**Document Structure:** Each record in the **SQLite3** database is represented as a document. The structure of the document is defined based on the requirements of the data to be stored.

**Scalability:** The **SQLite** database technology used allows for **horizontal scaling**, which ensures that the system can handle increasing volumes of data.

**Software:** Software is designed to handle requests from the system to access data from the **SQLite** database. The s/w are handled by the **android studio** in the development process to connect the project with the **cloud firebase.**

**SQLITE3**

SQLite is a popular open-source relational database management system that uses a file-based approach to store data. The file design in SQLite involves creating a database file that contains one or more tables, each of which consists of rows and columns.

**Database file:** This is the file that contains all the data for a SQLite database. It has a file extension of ".db" and can be accessed using a SQLite client or a programming language with SQLite bindings.

**Tables**: Tables are the primary storage unit in a SQLite database. Each table consists of one or more columns, which define the type of data that can be stored in each field, and one or more rows, which represent individual records.

**Schema:** The schema of a SQLite database describes the structure of the tables, including the name of each table, the name and data type of each column, and any constraints or indexes that apply to the data.

**Data types:** SQLite supports a wide range of data types, including integers, floating-point numbers, text, dates and times, and binary data.

**Primary key:** A primary key is a column or set of columns in a table that uniquely identifies each row. It is used to enforce data integrity and ensure that there are no duplicate records.

**Indexes:** An index is a data structure that speeds up the process of searching for data in a table. It is created on one or more columns and can improve performance for queries that filter, sort, or group data.

Overall, SQLite file design is flexible and can be adapted to a wide range of use cases, from small embedded systems to large-scale web applications. However, as with any database system, it is important to carefully consider the design of your tables, indexes, and schema to ensure that your data is stored efficiently and can be easily accessed and updated.

# INPUT AND OUTPUT DESIGN

# LOGIN

The administrator has to give their correct username and password to enter into next operations. The administrator can also change the password using forgot option.

# Input Design for Login Page

Actors : Administrator or Warden

Purpose : Unauthorized person cannot access the information.

**ADMIN**

**LOGIN**

In

valid d

If

Valid

User

id &

password

If id

&pas=id

User

Database

Not exist

Home

Page

# LOGIN PAGE

# NEW STUDENT:

Theadmin have the authorized access to add a new student to the hostel. The users have no access to this. The hostel admin more like the warden can add new student to the hostel as much as possible for the user to barrow and use.

# Input Design for New Student:

Actors : Administrator or Warden

Purpose : To add a student details in the hostel.

**NEW STUDENT**

Home

Page

In

valid

If

Valid

Student

Details

If a

ll

field fill

Store Student

Database

Fill all

filed

Added

successfull

New Student

**GUARDIAN INFORMATION:**

User of the hostel after being registered can borrow a students in the hostel. A user can store guardian of the student details.

# Input Design for Guardian info:

Actors : Administrator or Warden

Purpose : to store the guardian of the student details.

Student

information

Guardian

Student ID

Student

Database

ID

fee

payment

information

Info

**FEES PAYMENT**

User of the hostel after being registered can borrow a students in the hostel. A user can note the student fees details.

# Input Design for Fees payment

Actors : Administrator or Warden

Purpose : To store the student fees status if guardian pay fees notice that otherwise skip it.

guardian

information

pay

otherwise

Student ID

If

Pay fees

Student

Database

paid

Show Student

Details

info

**EXISTING STUDENT:**

User of the hostel after being registered can borrow a students in the hostel. A user can show the student details.

# Input Design for Existing Student:

Actors : Administrator or Warden

Purpose : To check the Student details is register or not and also show Student information

Home

Page

In

valid

If

Valid

Student ID

If

ID=std ID

Student

Database

S

td

ID

ID

Not found

Show Student

Details

Issue

# VISITORS:

User of the hostel after being registered can borrow a guardian can visit the student in the Hostel.

# Input Design for Visitor:

Actors : Administrator or Warden

Purpose : Visitors visite the students , id ,name and relationship is noted.

Visitors

In

valid

If

Valid

Student ID

If

studentr

Id Exist

Store visitor

Database

Student ID

Not found

Visite

student

visite

# OUTINGS:

The warden are note the out and in time of the student details.

# OUTPUT Design for Outings:

Actors: Administrator or Warden.

Purpose: Outings module is used to note the student out and in time of the hostel.

In time

Out time

Data Base

Outings

**MESS EMPLOYEE:**

Mess employee is used to maintain a employee details in hostel.

# Output Design for Delete Book:

Actors : Administrator or Warden

Purpose : Mess Employee is used to maintain a employee details in hostel..

emoloyee

Home Page

In

valid

If

Valid

Emp ID

If

Emp

Id Exist

Store Emp

Database

Wrong ID

exist

Show

Employee

Details

**MESS EMPLOYEE**

**WARDEN:**

Warden module is used to maintain a warden details in hostel

# Output Design for Warden Module:

Actors : Administrator or Warden

Purpose : Warden model is uded to

invalid

warden

Home Page

If

Valid

warden ID

If

warden

Id Exist

Store Warden

Database

ID not

exist

Show Warden

details

**WARDEN DETAILS**

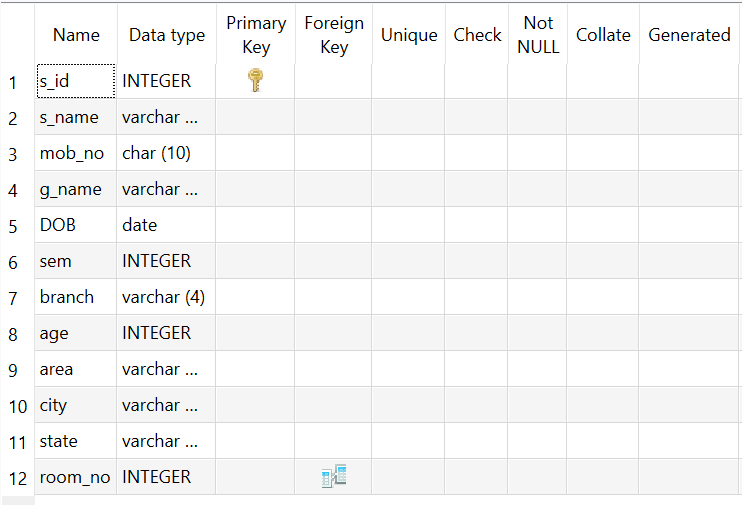
**DATABASE DESIGN**

**TABLE DESIGN**

**VARIOUS TABELS TO MAINTAIN INFORMATION**

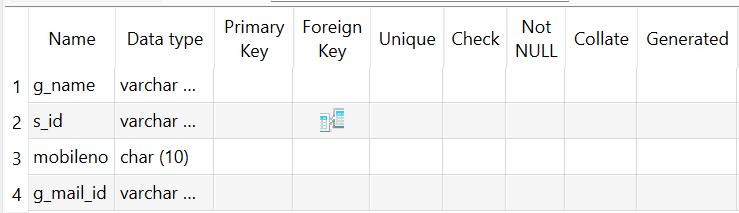
**STUDENT TABLE**

* Student table is used for storing an information about students



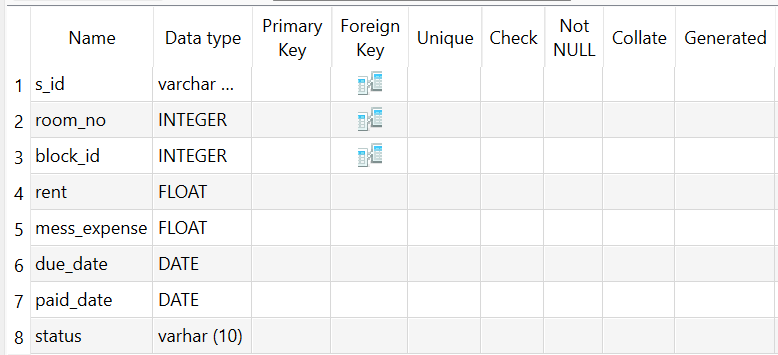
**GUARDIAN TABLE**

Guardian table is used for maintain guardian information



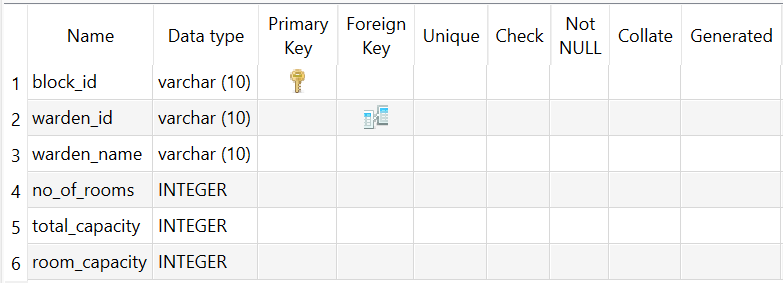
**FEES TABLE**

Fees Table is used for store the rent and mess fees of the student



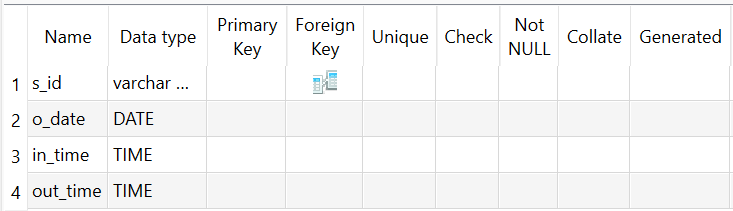
**BLOCK TABLE**

Block table is used for maintaining blocks of the hostel.



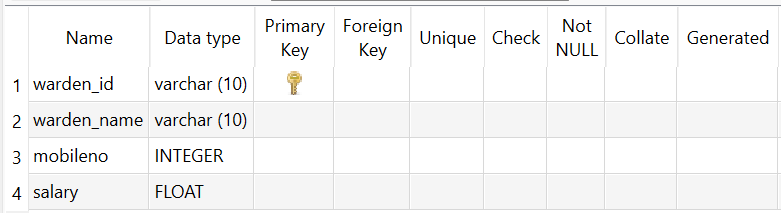
**OUTINGS TABLE**

Outing table is used for store and maintaining in and out time of the student



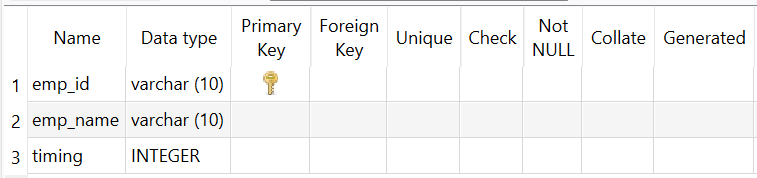
**WARDEN TABLE**

Warden table is used for storing an information about warden



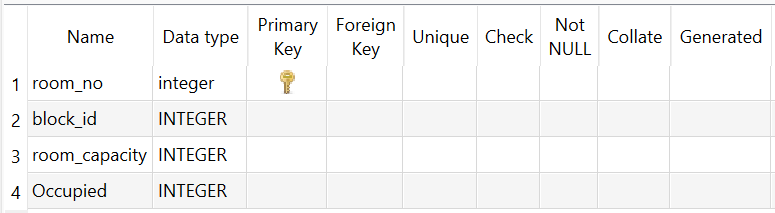
**MESS TABLE**

Mess table is used for storing mess employee information



**ROOM TABLE**

Room table is used for storing capacity and occupied of the rooms.



**SYSTEM DEVELOPMENT**

**INTRODUCTION**

The Developing of a hostel management system involves designing and implementing software that automates various tasks related to managing a hostel or dormitory. This system can streamline operations, enhance efficiency, and improve overall hostel management. Here are some key steps to consider when developing a hostel management system:

**Requirement Analysis:**

Understand the specific requirements and objectives of the hostel management system. Identify the key features and functionalities needed to support various operations, such as room allocation, check-ins/check-outs, billing, inventory management, and reporting.

**System Design:**

Create a system design that encompasses the architectural components, data structures, and interfaces. Consider the user interface (UI) design to ensure ease of use and a seamless experience for hostel administrators, staff, and residents.

**Database Design:**

Develop a database schema to store and manage data related to hostels, rooms, residents, bookings, payments, and other relevant information. Decide on the database management system (e.g., MySQL, PostgreSQL) that suits your requirements.

**User Management:**

Implement user management functionalities to differentiate between hostel administrators, staff, and residents. Provide authentication and authorization mechanisms to ensure secure access to the system.

**Room Allocation:**

Design a mechanism to manage room allocation efficiently. This can include features like room availability tracking, assigning rooms based on preferences or constraints, and managing multiple room types.

**Booking and Reservation:**

Develop functionalities for managing booking requests, reservations, and cancellations. Enable residents or authorized personnel to make reservations through the system, and ensure that real-time availability information is provided.

**Check-in and Check-out:**

Implement features to streamline the check-in and check-out processes. This can include capturing resident details, generating and printing registration cards or invoices, updating room status, and managing room keys.

**Billing and Payments:**

Create modules to manage billing and payments. Enable automatic calculation of charges based on room rates, additional services, and duration of stay. Integrate with payment gateways to facilitate online payments and generate invoices or receipts.

**Inventory Management:**

Include functionalities to track and manage inventory items such as toiletries, bedding, and kitchen supplies. Implement features for inventory tracking, reordering, and generating reports on stock levels.

**Reporting and Analytics:**

Develop reporting capabilities to generate various reports, such as occupancy rates, financial summaries, resident statistics, and inventory usage. Enable administrators to extract insights and make data-driven decisions.

**Integration and Scalability:**

Ensure the system can integrate with other existing systems, such as accounting or CRM software. Design the system to be scalable, allowing for future expansion and accommodating increasing numbers of hostels, rooms, and users.

**Testing and Deployment:**

Conduct comprehensive testing to identify and rectify any issues or bugs. Once testing is complete, deploy the system on the desired infrastructure, such as on-premises servers or cloud-based platforms.

**Maintenance and Support:**

Provide ongoing maintenance and support to ensure the system runs smoothly. Regularly update the system to address any security vulnerabilities, bugs, or feature enhancements.

It is important to note that the above steps provide a general outline, and the specific requirements of your hostel management system may vary based on factors such as the scale of the hostel, the target audience, and the desired functionalities.

**CHAPTER 4**

**SYSTEM TESTING**

**INTRODUCTION**

Testing is the set of activities that can be planned in advanced and s conducted systematically. Testing requires that the developer discard preconceived notions of the “correctness” of the software just developed and overcome a conflict of interest that occurs when errors are encountered.

**Testing principles are**

* All tests should be traceable to customer requirements
* Testing should be planned long before the testing begins
* Testing should begin “in the small” and progress towards testing “in the Large”.
* Exhaustive testing is not possible
* To be most effective, testing should be conducted by an independent third Party

**Testing objective are**

* Testing is the process of executing a program within the intent of finding an error.
* A good test case is one that has high probability of finding an as-yet-Undiscovered error.
* A successful test is one that un covers an as-yet-undiscovered error.

There are various testing strategies available to accommodate from low-level testing to high-level testing as discussed below.

**TEST PLAN**

Testing is the major quality control measure employed during software development. In the project, the first test considered is the unit testing. In this unit testing, each modules of the system are tested separately. This is carried out duringProgramming stage itself. Each module should work satisfactory as regard from the module.

After the entire module are checked independently and completed then the integration testing is performed to check whether there is any interface errors. Then those errors are verified and corrected.

And also the security test is performed to allow only authorized persons to this system. Finally, the validation testing is performed to validate whether the customer requirements are stratified are not,

**UNIT TESTING**

The unit testing is carried out on coding. Here different modules are tested against the specifications produced during design for the modules. Unit testing mainly focused first in the smallest and low level modules, proceeding one at a time. Each module was tested against required functionally and test cases were developed to test the boundary values.

Unit testing focuses verification effort on the smallest unit of software design the software component or module. The unit test focuses on the internal processing logic and data structures within the boundary of the component. This type of testing can be conducted in parallel for multiple components.

Each module was tested against required functionally and test cases were developed to test the boundary values.

**Existing student Module**

The user should enter Student id to search the Student information. Otherwise an error message is displayed to the user.

The user should select the accession number of the room to view the complete details. If the user not yet select then it shows the message to select the particular room number.

This system does not allow allocate room, if it is already occupied room

**INTEGRATION TESTING**

Integration testing is a systematic technique for consulting the software architecture while at the same time conducting test to uncover errors associated with interfacing. The objective is to take unit tested components and build a program structure that has been dictated by design.

This software contains four modules and each module is linked with one another by passing the data from one stage to another stage to perform the operation. All the modules are integrated and working properly.

**VALIDATION TESTING**

Validation testing is that validation succeeds when software functions in a manner that can be reasonably expected by the user. Validation testing begins after the culmination of integration testing, software is completely assembled as a package: interfacing errors have been uncovered and corrected.

The error detecting during this testing is

* Incorrect Function
* Input Condition Errors
* Database Error and Performance Error
* Initializing and performance Error

**Test Case**

The requirements collected from the customer at the time of requirements gathering is validated with the system and the requirements are satisfied.

The user requested to display the rack number and rack name of the books, journals and the project reports placed in the library is satisfied this system.

The user also asks for book reservation. It also satisfied by this system.

The user also wants to get the reports of the students and the staffs visiting the system, it also accomplished.

**SECURITY TESTING**

Security testing verifies that protection mechanisms built into a system will, in fact, protect it from improper penetration. The system security must, of course, be tested for invulnerability from flank or rear attack

**Test case**

The system provides authentication by means of validating the username and password. It won’t allow the user until he/she gave the correct password.

**WHITE-BOX TESTING**

White-box testing (also known as clear box testing, glass box testing, and transparent box testing and structural testing) tests internal structures or workings of a program, as opposed to the functionality exposed to the end-user. In white-box, testing an internal perspective of the system, as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths through the code and determine the appropriate outputs. This is analogous to testing nodes in a circuit, c.g. in-circuit testing (ICT).

While white-box testing can be applied at the unit, integration and system levels of the software testing process, it is usually done at the unit level. It can test paths within a unit, paths between units during integration, and between subsystems during a system- level test. Though this method of test design can uncover many errors or problems, it might not detect unimplemented parts of the specification or missing requirements.

**BLACK-BOX TESTING**

Black-box testing treats the software as a "black box", examining functionality without any knowledge of internal implementation. The tester is only aware of what the software is supposed to do, not how it does it. Black-box testing methods include: equivalence partitioning, boundary value analysis, all-pairs testing, state transition tables, decision table testing, fuzz testing, model-based testing, use case testing, exploratory testing and specification-based testing.

**DATA FLOW DIAGRAM**

Data flow diagram is a structured analysis tool for showing the movement of data through the different transformation or processes in the system. Named bubbles show the processes and data named arrows, entering or leaving the bubbles, represent flows. A rectangle represents a source or sinks and is a net originator or consumer of data. A source or sink is typically outside the main system study, DFD can be hierarchically organized which helps in partitioning and analyzing the large system, such system are called leveled DFD's.

**DFD SYMBOLS:**

In the DFD, there are four symbols

* A square defines a source (originator) or destination of system data
* 2An arrow identifies data flow. It is the pipeline through which the information flows.
* A circle or a bubble represents a process that transforms incoming data flow into outgoing data flows.
* An open rectangle is a data store, data at rest or a temporary repository of data

Process that transforms data flow.

Source or Destination of data

Dataflow

**CONSTRUCTING A DFD**

Several rules of thumb are used in drawing DFD's:

* Process should be named and numbered for an easy reference. Each name should be representative of the process.
* The direction of flow is from top to bottom and from left to right. Data traditionally flow from source to the destination although they may flow back to the source. One way to indicate this is to draw long flow line back to a source. An alternative way is to repeat the source symbol as a destination. Since it is used more than once in the DFD it is marked with a short diagonal.
* The names of data stores and destinations are written in capital letters. Process and dataflow names have the first letter of each work capitalized.
* When a process is exploded into lower level details, they are numbered.

A DFD typically shows the minimum contents of data store. Each data store should contain all the data elements that flow in and out. Questionnaires should contain all the data elements that flow in and out. Missing interfaces redundancies and like is then accounted for often through interviews.

**0-Level DFD**

A 0-level Data Flow Diagram (DFD) for a Library Administrator System provides a high-level overview of the system's processes and their interactions with external entities. It represents the system as a single process and shows the flow of data between the system and its external entities without delving into the internal details. Here's an explanation of the components typically included in a 0-level DFD for a Library Administrator System.

Outings

Warden

Login Details

Existing Student

New Student

Visitors

Mess Employee

Fees Renewal

**Zero Level DFD Hostel Administration System**

**1-Level DFD**

A 1st-level Data Flow Diagram (DFD) for a Hostel Administrator System expands on the 0-level DFD by breaking down the main process into more detailed sub-processes. It provides a more granular view of the system's functionalities and how data flows between different processes and entities.

Hostel Administration System

Show Student Details

Added Successfully

Added Successfully

Added Successfully

Successfully

Show fees Status

Show Employee Details

Add Student

Visitors

Outings

Fees Renewal

Mess Employee

Existing System

Warden

Show Warden Details

**First Level DFD Hostel Administrator System**

**2-Level DFD**

The second-level DFD provides more detail about the main functions of the system that were identified in the first-level DFD. The second-level DFD shows how data flows between processes, data stores, and external entities.

Admin

Add student

Existing Student

Visitors

Outings

Fees Renewal

Mess Employee

Warden

Forgot password

Password Changed

Manage module

Check Credential

Login to

System

Check role o of access