EASY LEAVE

A leave management software.

TABLE OF CONTENTS

INTRODUCTION AND OBJECTIVES

SYSTEM STUDY

SYSTEM ANALYSIS

SYSTEM DESIGN

DIAGRAMS

SYSTEM DEVELOPMENT

TESTING

SYSTEM IMPLEMENTATION

COST ESTIMATION OF THE PROJECT

SECURITY AND VALIDATION CHECK

FUTURE SCOPE OF SOFTWARE

INTRODUCTION AND OBJECTIVES

1.1 Introduction

Leave Management in an organization is all about managing the leave requests and maintaining a record of it. There may be multiple reasons to write a leave application or a leave letter. A leave letter can be written by a student to the principal, an employee to the manager or HR, a teacher to the principal, a student to the hostel warden, etc. When you want to take leave, you will have to seek permission from your higher authority. While writing a leave letter, it is very essential to mention the reason behind the leave request, irrespective of whether you are working or studying.

Reasons of absence can vary. Some of them can be Illness or any other medical needs, taking rest or recovery from illness, A marriage function, Birthday, Anniversary, or any other family functions, Travelling with family or friends and many more.

It's very troublesome and sometimes fault prone to handle all the leaves of the organization with large number of people. It's very difficult to maintain the Leave requests manually or through Paper based format. So, to overcome these problems, the organizations require a centred software system which eliminates the paper work based process.

Our EASY LEAVE MANAGEMENT SOFTWARE allows its user to apply leave to their higher authority easily. It stores and maintain the Leave records in a more efficient and Systematic way. Through this software the leave applier gets to know whether his leave request is approved or not. The applier and manager can have all the history of existing leaves. This software provides emergency leave features where the applier can take quick leave according to their situations. Also help to calculate the number of Leaves taken monthly or annually by employees or students. All the content of user and manager are secured with user id and password so that nobody can access it. When any organisation uses our software, there will two main section one section for applier and another section for leave manager with easy software interface.

1.2 Objectives of project

Using our leave management system gives numerous benefits such as:

➢ To get better adopted to the modern technology.

➢ To reduce the human work load.

➢ This software will save time of the user in applying leave.

➢ Stores all the records in one place.

➢ Human forgets things as the time passes, by using our software, they can have all the history records of leaves.

➢ Fast service of leave requests can be done through this software which replaces the manual process to Computerized format.

➢ Monitoring, providing approval and disapproval for leave requests and also managing the pending leave requests, such operations can be performed effectively using this software.

➢ Error prone can be eliminated using the Leave management Software.

➢ To make everything in Leave management easy and understand better.

1.3   Existing System

➢   In existing system Leave process is carried out manually.

➢   Users and manager can have the fear of losing records.

➢   There is no security for record.

➢   Managing the history of Leaves is tedious to maintain in Existing system.

➢   The HR needs to calculate total leaves taken in a month or annually of each member,

      leads to wasting of time and miscalculations.

➢   Requires more time as it involves the paper work and approval process.

➢   It takes time for the User to get to know about the approval or disapproval decision from

      the higher authority.

➢   User cannot ask for the quick leave.

➢   There can be chances of data duplication in manual work.

➢   Manager cannot manage all the leave of different department of organisation at once.

1.4   Benefit of system

➢   In proposed system the Leave requests are managed in computerized format.

➢   User gets to know about approval or disapproval in lesser time.

➢   Eliminates paper work as the process is carried through software.

➢   User can have all the record of past leave in their history section.

➢   Calculation of the leaves throughout the year or month of all the members can be carried

    out effectively as software itself does the calculations.

➢   There will be security to user data.

➢   It has a unique feature of filtering Approved, disapproved and Pending Leave list of

    Employees.

➢   Manager can manage leave all the department at one place by using our software.

➢   Proposed software has the quick leave for emergency situation.

➢   Data duplication can be avoided through this software.

1.5 Needs

I have designed the given system in SQL server to automate the process. The following steps that give the detailed information of the need of proposed system are:

➢ In the current system the records are handled manually. The manual handling of the record is time consuming and highly prone to error. To overcome the problem computerized system is to be undertaken.

➢ The basic need of this software is efficiency. The software should be efficient so that whenever an admin enters the details are updated in the database. The record will be useful for the other purpose.

➢ Some control of the project is under the hands of the authorized person who has the password to access it and illegal access is not supposed to deal with.

➢ Security is the main criteria for the software system. Since illegal access may corrupt the database. So, security has to be given in the project via SQL server connection.

1.6   Hardware and Software Requirements

HARDWARE SPECIFICATION

Software specification:

- ✓ Windows: - 8.1, 10, 11.
- ✓ Front end: - C#, Visual Studio 2019
- ✓ Back end: - Microsoft SQL Server 2019
- ✓ Frame Work: - .Net Framework 4.7.2
- ✓ Report: - Crystal Report

Hardware specification:

- ✓ RAM: - 2 GB
- ✓ Processor: - intel Pentium or above.
- ✓ HHD or SSD: -50 GB minimum.
- ✓ Monitor: - Good monitor with 50% RGB colour gamut.
- ✓ Printer: - 80 columns colour printer.

Front End

C SHARP

C# is a general-purpose high-level programming language supporting multiple paradigms. C# encompasses static typing, strong typing, lexically scoped, imperative, declarative, functional, generic, object-oriented, and component-oriented programming disciplines.

C# is an object-oriented, component-oriented programming language. C# provides language constructs to directly support these concepts, making C# a natural language in which to create and use software components. Since its origin, C# has added features to support new workloads and emerging software design practices. At its core, C# is an object-oriented language. You define types and their behaviour.

C# programs run on .NET, a virtual execution system called the common language runtime (CLR) and a set of class libraries. The CLR is the implementation by Microsoft of the common language infrastructure (CLI), an international standard.

Visual Studio

An *integrated development environment* (IDE) is a feature-rich program that supports many aspects of software development. The Visual Studio IDE is a creative launching pad that you can use to edit, debug, and build code, and then publish an app. Over and above the standard editor and debugger that most IDEs provide, Visual Studio includes compilers, code completion tools, graphical designers, and many more features to enhance the software development process.

- First, you have to download and install the Visual Studio. For that, you can refer to Downloading and Installing Visual Studio 2019. Don't forget to select the .NET core workload during the installation of VS 2019. If you forget then you have to modify the installation.
- You can see a number of tool windows when you will open the Visual Studio and start writing your first program as follows:
- **Code Editor:** Where the user will write code.
- **Output Window:** Here the Visual Studio shows the outputs, compiler warnings, error messages

.NET and .NET Framework

.NET is a developer platform made up of tools, programming languages, and libraries for building many different types of applications.

There are various implementations of .NET. Each implementation allows .NET code to execute in different places—Linux, macOS, Windows, iOS, Android, and many more.

1. **.NET Framework** is the original implementation of .NET. It supports running websites, services, desktop apps, and more on Windows.
2. **.NET** is a cross-platform implementation for running websites, services, and console apps on Windows, Linux, and macOS. .NET is open source on GitHub. .NET was previously called .NET Core.
3. **Xamarin/Mono** is a .NET implementation for running apps on all the major mobile operating systems, including iOS and Android.

.NET Standard is a formal specification of the APIs that are common across .NET implementations. This allows the same code and libraries to run on different implementations.

Crystal Report

SAP Crystal Reports can help you analyse your data by creating richly formatted, pixel-perfect, and multipage reports from virtually any data source, delivered in over a dozen formats.

SAP Crystal Reports is a BI tool for generating analytical reports from SAP and other non-SAP data sources like Oracle, SQL Server, MySQL, XML Data Source, Microsoft Excel, etc. The knowledge of this tool helps businesses to develop advanced level reports and take accurate and profitable business decisions based on these reports.

The Report Design Canvas helps you to design the structure of your report. You can include different elements like charts, test, data objects in the reports.

Rules:

Rules option displays a rule above the report canvas. However, the tool allows you to change the unit of measurement by changing the unit option on the design canvas tab.

Back End

Microsoft SQL server

What is SQL?

- SQL stands for Structured Query Language
- SQL lets you access and manipulates Databases
- SQL is an ANSI (American national Standards Institute) standard

What can SQL do?

- SQL can execute queries against a database
- SQL can retrieve data from a database
- SQL can insert records in a database
- SQL can update records in a database
- SQL can delete records from a database
- SQL can create a new database
- SQL can create new tables in database
- SQL can create stored procedures in a database
- SQL can create view in a database
- SQL can set permission on tables, procedure, and views

RDBMS

RDBMS stands for relational database management system. RDBMS is the basis for SQL, and for all the modern database system such as MS SQL Server, IBM DB2, Oracle, MySQL, and Microsoft access. The data in RDBMS is stored in database object called tables. A table is a collection of related data entries and it consists of columns and row.

Some of the most important SQL Commands

- **SELECT**- extracts data form a database
- **UPDATE**- updates data in a database
- **DELETE**- delete data from a database
- **INSERTINTO**-inserts new data into a data base
- **CREATEDATABASE**- create a new database
- **ALTERDATABASE**-modifies a database
- **CREATETABLE**- create a new table
- **ALTERTABLE**-modifies a table
- **DROPTABLE**-delete a table
- **CREATEINDEX**-create index (search key)
- **DROP INDEX**-delete an index

Microsoft SQL Server 2019: includes a complete set of graphical tool and command line utilities that allow users, programmer, and administrator to increase their productivity. The step-by-step tutorials listed below, help you learn to get the most out of SQL Server tools so you can work efficiently, right from the start.

The following table describes the topic in this section. Click a link to start a tutorial.

To open SQL Server Management Studio

1. On the Start menu, point to all programs, point to Microsoft SQL in Server 2005, and then click SQL Server management Studio.
2. In Connect to Server dialog box, verify the default settings, and then click connect. To connect, the Server name box must contain the name of the computer where SQL Server is installed. If the Database Engine is a named instance name in the format

   <Computer name>\<instance name>

Create a new database in SQL Server 2005. Start Management studio as administrator by right clicking the icon and choosing "run as administrator". Connect to the database engine with username/password. Right click the folder that says "Databases", and choose "New Database".

SYSTEM STUDY

2.1 Preliminary Investigation

System development, a process consisting of two major steps of system analysis and design start when management or sometimes system development personnel feel that a new system or an improvement in the existing system is required. The system development life cycle is classically thought of as the set of activities that analyst, designers and user carry out to develop and implement an information system. The system development life cycle consists of the following activities:

- Preliminary investigation
- Determination of system requirements
- Design of system
- Development of software
- System testing
- Implementation, evaluation, and maintenance

A request to take assistance from information system can be made for many reasons, but in each case someone in the organization initiates the request is made, the first system activity the preliminary investigation begins. This activity has three parts:

1) Request clarification

2) Feasibility study

3) Request approval

Request clarification: Many requests from employees and users in the organizations are not clearly defined, therefore it becomes necessary that project request must be examined and clarified properly before considering systems investigation

2.2 System Development Life Cycle

Systems are created to solve problems. One can think of the systems approach as an organized way of dealing with a problem. In this dynamic world, the subject System Analysis and Design (SAD) mainly deals with the software development activities.

Defining a System

A collection of components that work together to realize some objective forms a system Basically there are three major components in every system, namely input, processing and output.

In a system the different component is connected with each other and they are interdependent. For example, human body represents a complete natural system. We are also bound by many national systems such as political system, economic system, educational system and so forth. The objective of the system demands that some output is produced as a result of processing the suitable inputs.

System Life Cycle

System life cycle is an organizational process of developing and maintaining systems. It helps in establishing a system project plan, because it gives overall list of processes and sub-processes required for developing a system

System development life cycle means combination of various activities. In other words we can say that various activities put together are referred as system development life cycle. In the system Analysis and design terminology, the system development life cycle means software development life cycle. A system analysis is a separation of a substance into parts for study and their implementation and detail examination.

Before designing any system, it is important that the nature of the business and the way it operates are clearly understood. The detailed examination provides the specific data required during designing in order to ensure that all the client requirements are fulfilled. The investigation or the study conducted during the analysis phase is largely based on the feasibility study. Rather begins during the feasibility study.

One of the main causes of project failures is inadequate understanding, and one of the main causes of inadequate understanding of the requirements. Analysis requirements us to recall the objectives of the project and consider following three requirements:

- ✓ What type of information is required?
- ✓ What are the constraints on the investigation?
- ✓ What are the potential problems that may make the task more difficult?

2.3 Feasibility Study

The basic premise of system analysis is being done here. The primary goal of the system is to do identify problems and determine how they can be solved with the computer system. In formal SDLC methodologies, the first step in system analysis is system feasibility study. A feasibility study is the quick examination of the problems, goals, expected cost of the system The objective is to determine whether the problem is reasonably solved with a computer system. In some cases, maybe there is a better alternative or perhaps is simply short-term annoyance and will gradually disappear. In other cases, the problem may turn out to be more complex that was thought and involves users across the company. Also, some problems may not be solvable with today's technology.

In any case, you need to determine the scope of the project to gain the better idea of cost, benefits, and objectives. The feasibility study is typically written so that nonprogrammers can easily understand it. It is used to-sell to the upper management and as a starting point for the next step. Additionally, it is used as a reference to keep the project on track, and to evaluate the progress of project team. Is the project cost effective or there is a cheaper solution? Will the proposed system improve the operation of the bank, will complicating factors prevent it from achieving its goals? Does the technology exist and does the firm have the staff to make the technology work? When the proposal is determined to be feasible, the term leaders are appointed and a plan and schedule are created. The schedule contains a detailed listing of what parts of the project are completed at each time. Of course, it extremely difficult to estimate the true cost and completion dates. Nonetheless, the schedule is an important tool to evaluate the status of the project and the progress of the team.

Steps in feasibility analysis are:

- Identify deficiency by pinpointing, missing functions, unsatisfactory, performance, excessive cost of operations.
- Set goals to remove these deficiencies.
- Goals must be quantified, realizable within the constraints of an organization, broken down into sub goals agreeable to all concerned.
- Set goals not only to remove deficiencies but also to effectively meet competition. For instance, goals must be based on what competitors do.

2.4 Economic Feasibility

Economic analysis is the most frequently used technique for evaluating the effectiveness of a proposed system. More commonly known as cost/benefit analysis, in this procedure we determine the benefits and savings that are expected from a proposed system. During the feasibility study phases, broad alternatives solutions are examined. For each alternate solution the cost and benefits have to be examined before designing of the alternatives.

Broad solutions will be consisting of:

- Specifications of information to be made available by the system.
- Description of what will be done manually and what the computer will do.
- Specification on new computing equipment needed or specification of expansion of an existing computer.

Cost and Benefit Analysis

Developing an IT software is an investment. After developing the software, it provides the profits to the organization. Profits can be monetary or in form of an improved working environment. However, it carries risk because in some cases an estimate can be wrong and the project might not actually turn out to be beneficial

Cost benefits analysis help to give management a picture of the cost, benefits and risks. It usually involves comparing alternate investments. Cost, benefits determine the benefits and saving that are expected from the system and compares them with the expected cost. In performing cost and benefit analysis it is important to identify cost and benefits factors Cost and benefits can be categorized into the following categories.

1) **Development Cost:** Development costs are the costs that are incurred during the development of the system. It is one time investment

2) **Operating Cost:** Operating cost are the expenses required for the day to-day running of the system. As operating cost are wages, supplies and overheads.

3) **Hardware/Software Cost:** It includes the cost of purchasing or leasing of computes and its peripherals. A software cost involves required software cost.

4) **Personnel Cost:** It is the money spent on the people involved in the development of the system

5) **Facility Cost:** Expenses that are incurred during the preparation of the physical site where the system will be operational. These can be wiring, flooring, acoustics, lighting, and air-conditioning.

6) **Supply Cost:** These are available coasts that are very proportionately with the amount of use of paper, ribbons, disks and others

Benefits

- ✓ We can define benefits as
  - Profit or Benefit= Income-Cost
- ✓ Benefits can be occurred by
  - Increasing income or
  - Decreasing costs or Both

My proposed project to Synergy Tech does everything those 3-4 employees or accountants are currently doing on paper work except on a computer.

Due to this factor, if the organizational goes ahead with my project, they would not need any personnel, and their costs of sustaining the organization go down radically, the software itself requires minimal memory to run as files are stored in a very defragmented manner and can easily be moved around as well as zipped, to preserve even more space. Hence, funds spent on management storage, are almost trifling, will not pose a problem in the future while operating.

2.5 Technical Feasibility

Today, very little is technically impossible. Consequently, technical feasibility looks at what is practical and reasonable technical feasibility addresses three major issues:

1. Is the proposed technology or solution practical?
2. Do we currently possess the necessary technology?
3. Do we possess the necessary technical expertise, and is the schedule reasonable?

Is the proposed technology or solution practical?

The technology for any defined solution is normally available. The question whether that technology is mature enough to be easily applied to our problems. Some firms like to use state-of-the-are technology, but most firms prefer to use mature and proven technology. A mature technology has a larger customer base for obtaining advice concerning and improvements.

Do we currently process the necessary technology?

Assuming the solutions required technology is practical, we must next ask assuming the solution's required technology is available, we must ask if we have the capacity. For instance, will our current printer be able to handle the new report and form required of a new system? If the answer to any of these questions is no, then we must ask ourselves, can we get this technology? The technology may be practical and available, and, yes, we need it. But we simply may not be able to afford it at this time. Although this argument borders on economic feasibility, it is truly technical feasibility. If we can't afford the technology, then the alternative that requires the technology is not practical and is technically infeasible.

2.6 Operational Feasibility

It is mainly related to human organization and political aspects. The points to be considered are:

- What changes will be brought with the system?
- What organizational structures are distributed?
- What new skills will be required? Do the existing staff members have these skills?
- If not, can they be trained in due course of time?

Generally, project will not be rejected simply because of feasibility study we appointed a small group of people who are familiar with information system techniques, who understands the parts of the business that are relevant to the project and are skilled in system analysis and design process Feasibility Report.

After studying the feasibility study of the project we can came to the following points, these results may change according to further analysis and design.

PROJECT NAME: Easy leave management system.

Definition of Problem or Opportunity

We have to make a computerize system (software) to make the working of companies easy and efficient so that software will replace the manual work with automated computerized process.

A system analysis is a separation of a substance into parts for study and their implementation and detailed examination.

Before designing any system, it is important that the nature of the business and the way it currently operates are clearly understood. The detailed examination provides the specific data required during designing in order to ensure that all the client's requirements fulfilled. The investigation or the study conducted during the analysis phase is largely based on the feasibility study rather it would not be wrong to say that the analysis and feasibility phases overlap High-level analysis begins during the feasibility study. Through analysis is represented as one phase of the system development life cycle (SDLC), this is maintenance Even after successful implementation of the system, analysis may play its role for periodic maintenance and up gradation of the system.

One of the main causes of project failures is inadequate understanding, and one of the main causes of inadequate understanding, of the requirements is the poor planning of system analysis. Analysis requires us to recall the objectives of the project and consider following three questions:

- What type of information is required?
- What are the constraints on the investigation?
- What are the potential problems that may the task more difficult?

Keeping the above question in mind and considering the survey conducted to determine the need of the system, the total system was designed and can be described as under. The three major parts of the system are:

Providing Information

The system is effectively used to provide large variety of information to the interested Admin the major purpose of the system is to easily provide student details, course details, fees details with quick update to latest modification in the records. This thing is not at all possible in printed material, which are updated only once a few weeks. The system itself works as a information provider for management.

Constraints: After the objectives were clear during the analysis phase, it was essential to understand the constraints in order to plan and avoid problems arising during detailed analysis.

Technology: The admin may be committed to a particular hardware or software solution. The software required in this case is, complete net version and SQL server. Budget- if budget is a real constraint, the budget of the new system proposed would be constantly compared with that of the existing system or any alternatives solution. In this case during the economic feasibility, it has been clearly proved that the new system is definitely more feasible than the alternative solution possible. Organization must implement a system which saves the effort, also it provides an easy method for customer to get receipts on time. Scope- what is the area under investigation in this project? What are the boundaries of the system? What is the existent of possible usage of the new system?

Environmental Analysis: The external entities of an organization are its student or any individual staffs.

SYSTEM ANALYSIS

3.1 Importance of Easy Leave Management System

There are several attributes in which the computer-based information works Broadly the working of computer system is divided in to two main groups.

- Transaction System
- Decision Support System

Transaction System: A transaction system is a record of some well-defined single and usually small occurrence in a system. Transactions are input into the computer to update the database files. It checks the entering data for its accuracy. This means that numeric data appears in numeric field and character data appears in character field. Once all the checks are made, transactions are used to update the database. Transactions can be inputted in on-line mode or batch mode. In online mode transactions are entered and updated into the database instantaneously. In batch mode, transactions are collected into batches, which may be held for a while and inputted later.

Decision Support System: It assists the user to make analytical decision. It shows the various data in organized way called analysis. This analysis can be made to speedy preferences and help in making decisions. Computer System works out best with record maintenance. It will tell you which customer should pay pending reports/statements. It will also help to search the information about a particular person by simply entering his telephone number. User can store information as per requirement which can be used for comparisons of other reports.

3.2 Principles of System Analysis

Principles

- Understand the problem before you begin to create the analysis model.
- Develop prototypes that enable a user to understand how human machine interactions will occur.
- Record the origin of and reason for every requirement.
- Use multiple views of requirements like building data, functions and behavioural models.
- Work to eliminate ambiguity.

A Complete Structure

The limited time and resources have restricted us to incorporate, in this project, only the main activities that are performed in news sites, but utmost care has been taken to make the system efficient and user friendly.

For the optimum use of practical time, it is necessary that every session is planned Planning of this project will include the following things:

- Topic understanding
- Modular Break-Up of the system
- Processor Logic for Each Module
- Database Requirements

Topic Understanding

It is vital that the field of software as introduced in the project may be totally a new field. So as soon as the project was allocated to me, I carefully went through the project to identify the requirements of the project.

Modular Break-Up of the System

- Identify the various Modules in the System.
- List Them in The Right Hierarchy.
- Identify Their Priority of Development
- Description of the Modules.

3.3 Methods Used for Gathering Information

The methods used for gathering information about existing system are as followed

- Review of records
- Observation of the functioning system
- Interviews
- Questionnaires

In order to create an information and practical system, a system analyst would have to have some kind of way to view the current system. Receiving feedback on what can be done to improve the current system, and how much the current system is acceptable to the users.

Requirement Analysis

The main part of the problem is to obtain a clear understanding of the needs of user and what exactly are desired from the software. It is used for specifying the requirement.

Fact Finding Tools

After obtaining the background knowledge, I began to collect data on the existing system's output, input and costs. The tools used in data collection/ information gathering are: Review of the written Documents On-site observation Interviews Questionnaires.

Review of the Written Documents

In this phase we analysed all the documents like the day books, customer registration, invoice generation slip, course details broachers, accounts etc. All these things describe the format and functions of the current system included in most manuals are system requirement that help determine how these various objectives are met.

The page is one of the most important sources through which I draw some conclusions like:

- Who use the page(s)? How important are they to user?
- Do the pages include all the necessary information?
- What item should be added or deleted?
- How readable and easy to follow is the page?
- How does the information in the page help other user to make better decision?
- What other uses does the page offer the user area?

By analysing all the details, we draw a conclusion that what are the merit and De-merit of the

current phase but above all there are some problems with some onsite observations that one

analyst must face during analysis like:

Review of Documents on Site Observation, Interview Questionnaires, Data Organization,

Info Gathering Tools

- Take long time and get inefficient result
- Attitude and motivation of the subject cannot be readily observed.
- Observations are subject to error.
- In a complex situation it can be very time consuming.

SYSTEM DESIGN

System Design

The design document that we will develop during this phase is the blueprint of the software. it describes how the solution to the customer problem is to be built Since solution to complex problems isn't usually found in the first try, iteration is most likely required. This is true for design as well. For this reason, any design strategy, design method, or design language must be flexible and must easily accommodate changes due to iteration in the design any technique or design needs to supports and guide the partitioning process in such way that the resulting sub-problems are as dependent as possible from each other and can combined easy combination of their solution reduce the complexity of the problem. This is the objective of the partitioning process. Partitioning or decomposition during design involves three types of decisions:

- Define the boundaries along which to break
- Determine into how money pieces to break
- Identify the proper level of detail when design should stop and implementation should start.

Basic design principles that enable the software engineer to navigate the design process suggest set of principles for software design, which have been adapted and extended in the following list. It is free from the surfer from tunnel vision. A good designer should consider alternative approaches, judging each based on their requirements of the problem, the resources available to do the job. The design should be traceable to the analysis model Because a single element of the design model often traces to multiple requirements, it is necessary have means for tracking how requirements have been satisfied by the design model. The design should not repeat the same thing. Systems are constructed using set of which have likely been encountered using set of design patterns, many of which have likely been encountered before. These patterns should always be chosen as an alternative to reinvention. Time is short and resources are limited! Design time should be invested in representing truly new ideas and integrating those patterns that already exist. The design should minimize the intellectual distance between the software and the problem as it exists in the real world. That is, the structure of the software of the problem domain. The design should exhibit uniformity and integration. A design is uniform if it is appearing that one person developed the entire thing rules of style and format should be defined for a design term before design work begins. Once the designer is satisfied with the design he has produced, the design is to be precisely specified in the form of a document. To specify the design, specification languages are used. Producing the design specification is the

ultimate objective of the design phase. The purpose of this design document is quite different from that of the design notation, whereas a design represented using the design notation is largely to be used by the designer, a design specification has to be so precise and complete that it can be used as a basis of further development by other programmers. Generally, design specification uses textual structures, with design notation helping in understanding.

4.1 Physical Design

The design phase focuses on detailed implementation of the system recommended in the feasibility. Emphasis is on translating performance specifications. The design phase is a transition from user-oriented document to programmer-oriented document.

A. Design Methodology: Design methodology is a way to transform the art of system analysis and design into an engineering type discipline. It explains the relationship amongst various modules and programs within the system. It standardizes the approach to analysis and design, simplifies design by segmentation, and improves documentation and subsequent maintenance and enhancements. The following structured diagram can appropriately represent the relationship between various modules.

B. Design Overview: In analysing the present system a great deal of information was collected during the investigation and feasibility phases through list of problem and requirements, interview report, questionnaires, onsite observation, manual and determining potential solutions. It is important to record this information in an unambiguous, concise manner which will be clear and accessible to others. And which can be used by other analyst and designers involved in developing the system. Structured techniques help us to record the information in this way, using diagrams and minimum amount of the next. Structured analysis is a set of technique and graphical tools that allow the analyst to develop a new kind of system specification that are easily understandable to the user. The traditional approach of organizing data through flow chart support future developments and simplify communication with the user but focus on the cost/benefits and feasibility analysis, project management, hardware and software selection, and personal consideration in contrast, structured analysis considers new goals and structured tools for analysis, which provide the basis for design and implementation.

C. Process Modelling: System design goes through two phases of development logical and physical Logical implementation represented by Data flow diagram shows the logical flow of a system and defines the boundaries of the system it describes the input(source),

output(destination), data bases (data stores), and procedures (data flow). All in the format that needs the user's requirements. The logical implementation of the whole project can be represented as under through data flow diagram (DFD).

4.2 Data Flow Diagram

Data flow diagrams are the most commonly used way of documenting the processing of the candidate system. As their name suggest they are a factorial way of representing the flow of data into, around and the out of the system. They are easily understandable and are less prone to misinterpretation than textual description A complete set of DFDs provides a compact top-down representation of the system, which makes it easier for the user and the analyst to envisage the system as whole DFDs are constructed using four major components.
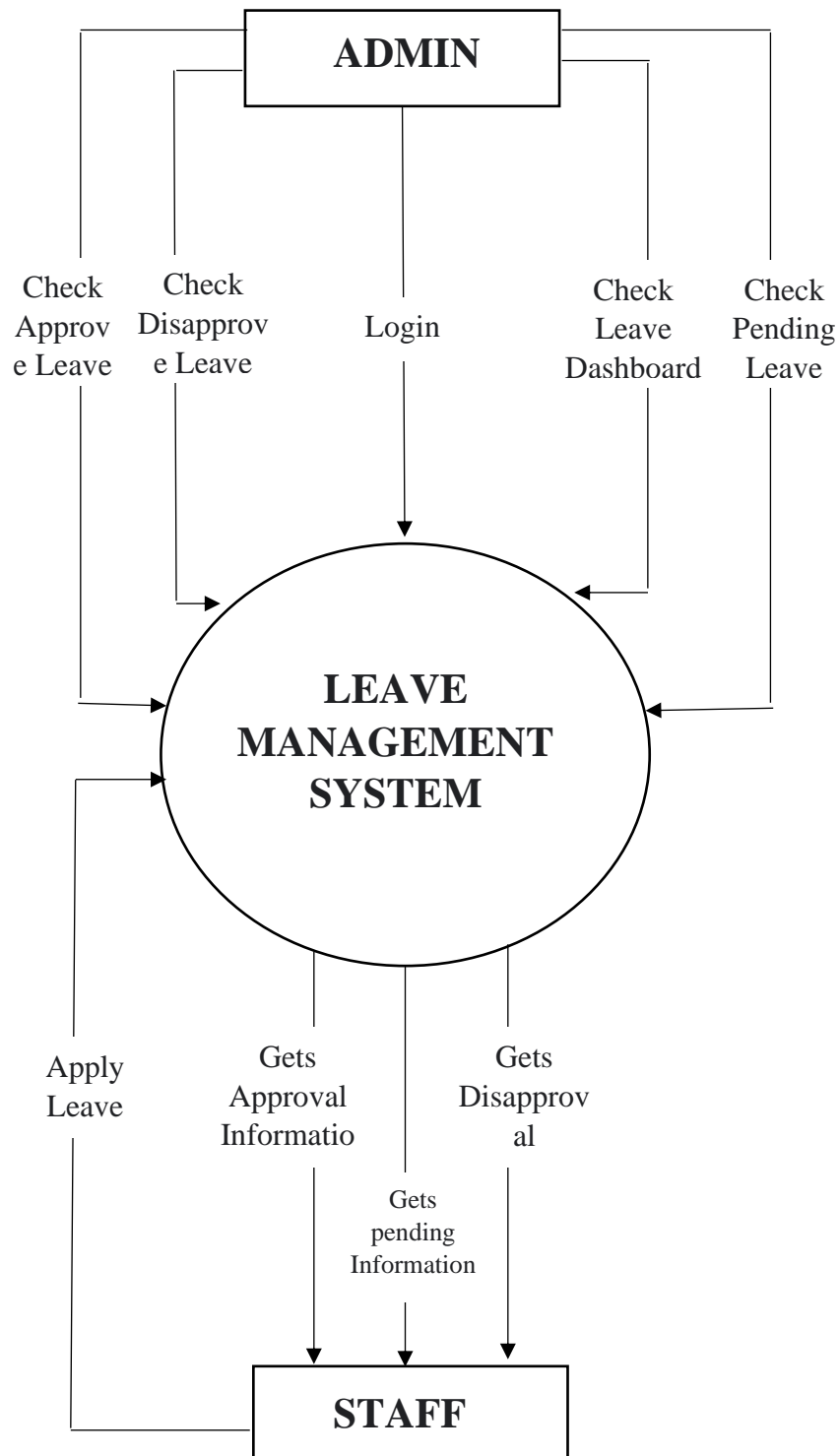
- **External Entities:** represents the sources of the data that enter the system or the recipients of the system that leaves the system. For example-passenger is the usual receiver of the information and supplier of data during form filling
- **Data Stores:** represent the stores of the data within the system example. Computer files, database or in the manual system files Etc. Data stores cannot be linked directly by data flows either to each other or to external entities without an intervening process to transform them.
- **Processes:** represent activities in which data is manipulated by being stored or retrieved or transformed in some way process names are generally unambiguous and convey as much meaning as possible without being too long. Example: verify data, acquired time schedule etc.
- **Data Flow:** represents the movement of the data between other components. Data flow diagrams are used to describe how the system transforms information. They define how information is processed and stored identify how the information flows through the processes.

DIAGRAMS

4.3 Context level data flow diagram
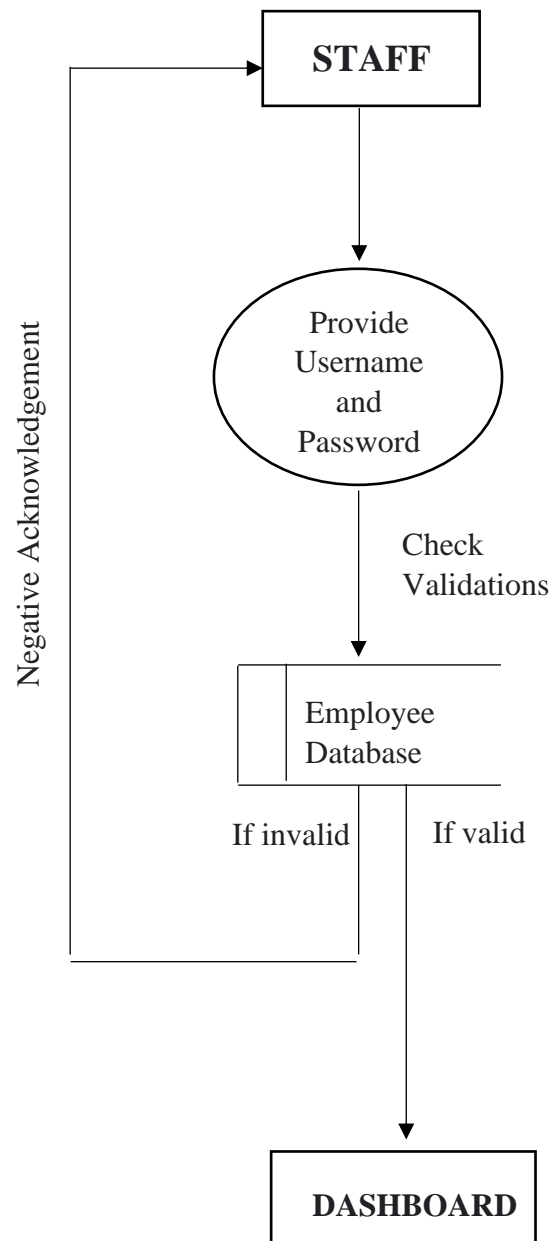
0-level DFD

It is also known as a context diagram. It's designed to be an abstraction view, showing the system as a single process with its relationship to external entities. It represents the entire system as a single bubble with input and output data indicated by incoming/outgoing arrows.
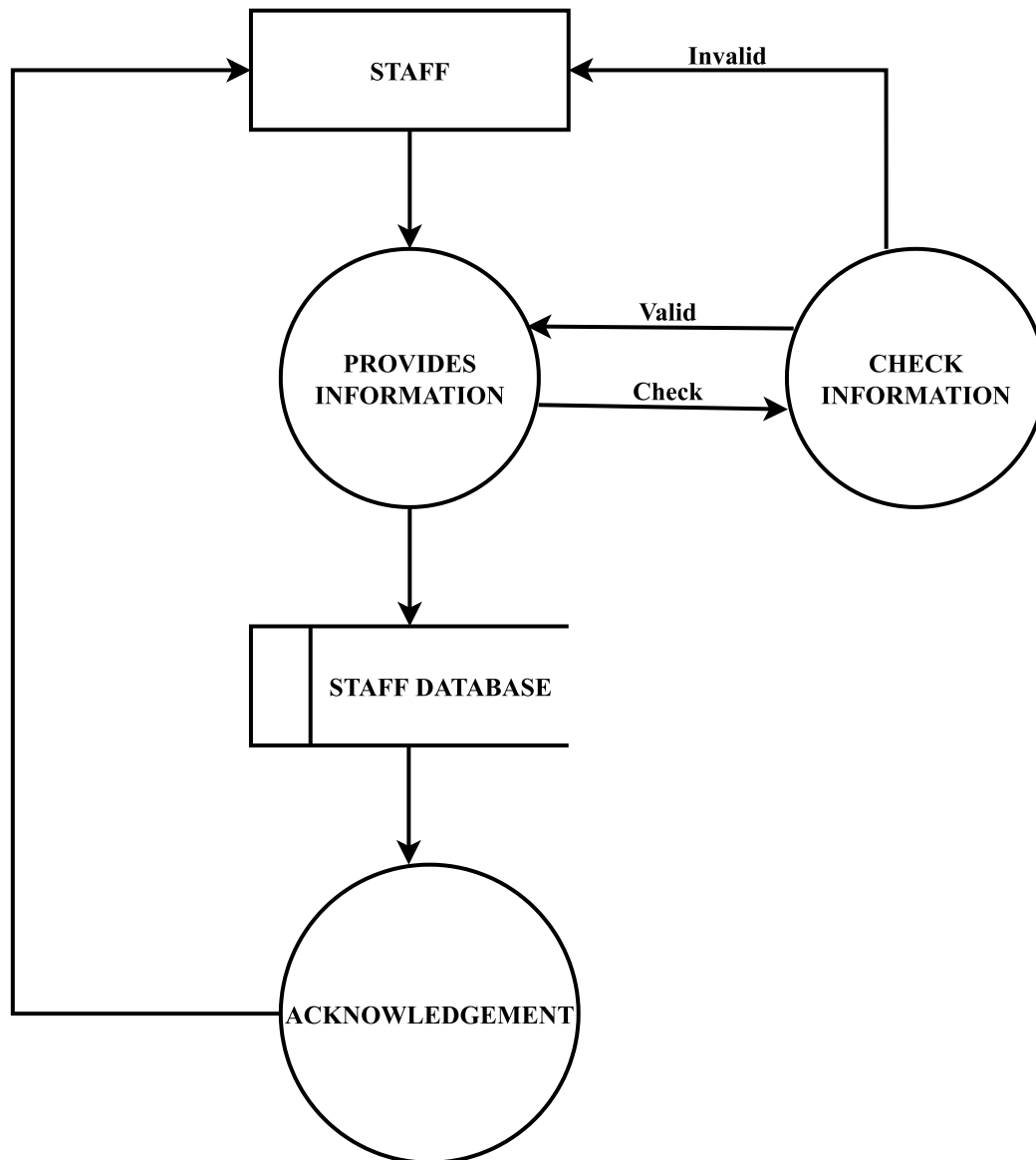
ADMIN

Check Approve Leave

Check Disapprove Leave

Login

Check Leave Dashboard

Check Pending Leave

LEAVE MANAGEMENT SYSTEM

Apply Leave

Gets Approval Informatio

Gets pending Information

Gets Disapproval

STAFF

Level 1 data flow diagram

1.0-Level DFD For Student Registration

In 1-level DFD, the context diagram is decomposed into multiple bubbles/processes. In this level, we highlight the main functions of the system and breakdown the high-level process of 0-level DFD into subprocesses.
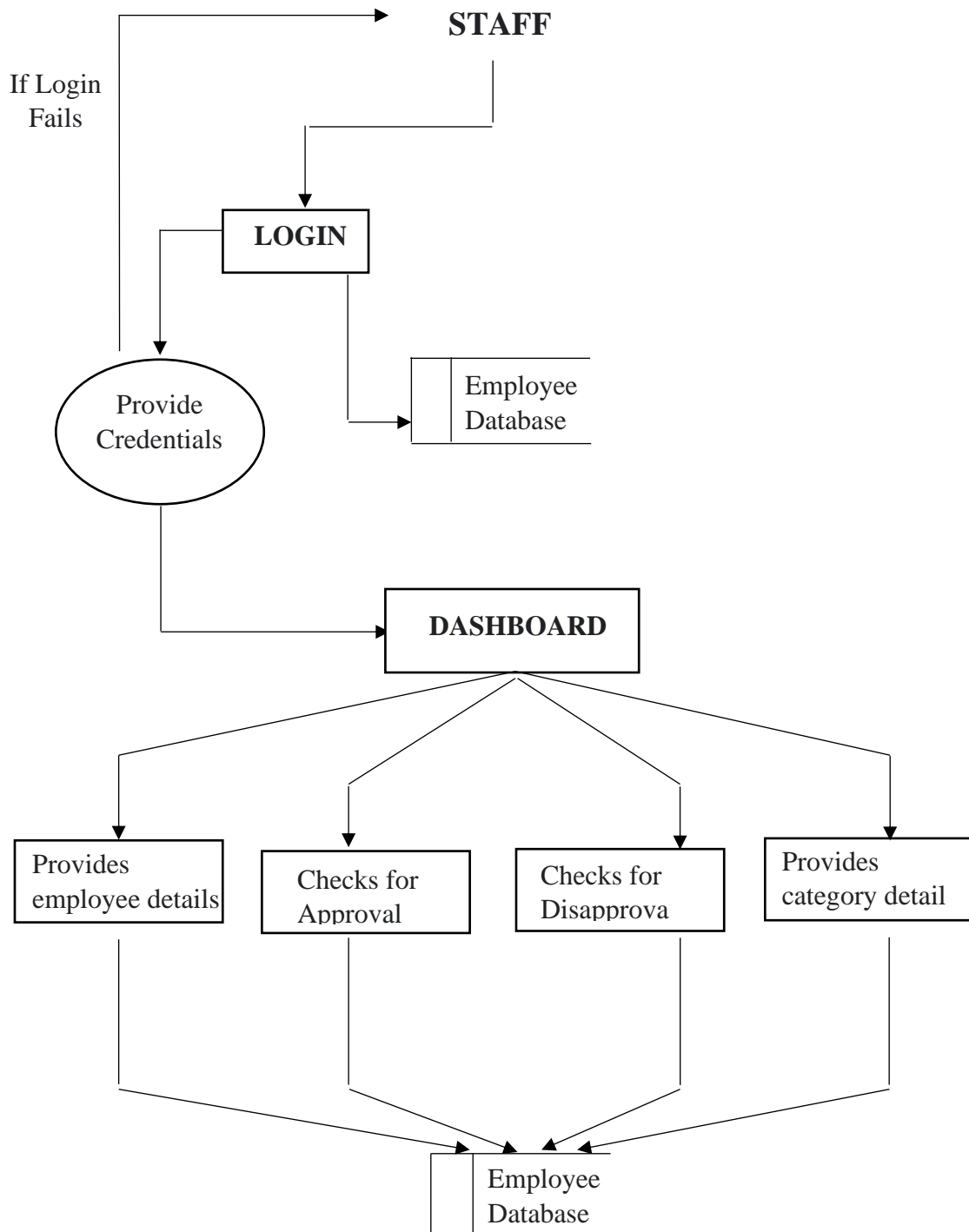
```
                              ┌──────────────┐
                              │    STAFF     │
                              └──────────────┘
                                     │
                                     ▼
                              ╭──────────────╮
         Negative             │   Provide    │
         Acknowledgement      │   Username   │
                              │     and      │
                              │   Password   │
                              ╰──────────────╯
                                     │
                                  Check
                                  Validations
                                     │
                                     ▼
                              ┌──────────────┐
                              │   Employee   │
                              │   Database   │
                              └──────────────┘
                        If invalid      If valid
                                     │
                                     ▼
                              ┌──────────────┐
                              │  DASHBOARD   │
                              └──────────────┘
```
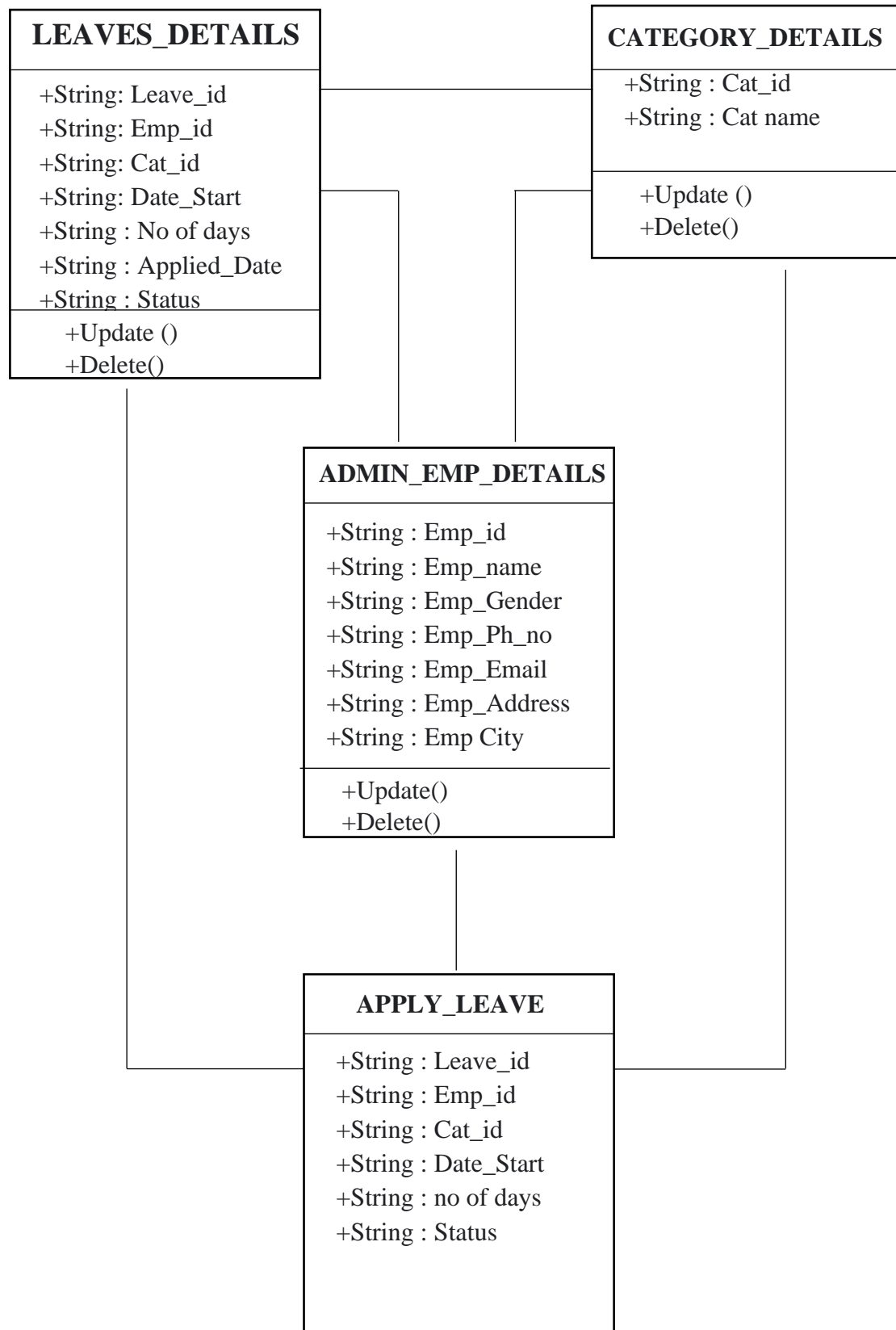
1.1-Level DFD For Staff Registration

```
                        ┌──────────────────┐         Invalid
              ┌────────▶│      STAFF       │◀──────────────────┐
              │         └──────────────────┘                   │
              │                  │                              │
              │                  ▼                              │
              │         ╭──────────────╮      Valid     ╭──────────────╮
              │         │   PROVIDES    │◀──────────────│    CHECK     │
              │         │  INFORMATION  │     Check     │ INFORMATION  │
              │         │               │──────────────▶│             │
              │         ╰──────────────╯                ╰──────────────╯
              │                  │
              │                  ▼
              │         ┌──────────────────┐
              │         │  STAFF DATABASE  │
              │         └──────────────────┘
              │                  │
              │                  ▼
              │         ╭──────────────────╮
              └─────────│ ACKNOWLEDGEMENT  │
                        ╰──────────────────╯
```

Level 2 data flow diagram

2-level DFD

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

STAFF

If Login
Fails

LOGIN

Provide
Credentials

Employee
Database

DASHBOARD

| Provides employee details | Checks for Approval | Checks for Disapprova | Provides category detail |

Employee
Database

4.4 Class diagram

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application.

LEAVES_DETAILS

+String: Leave_id
+String: Emp_id
+String: Cat_id
+String: Date_Start
+String : No of days
+String : Applied_Date
+String : Status

+Update ()
+Delete()

CATEGORY_DETAILS

+String : Cat_id
+String : Cat name

+Update ()
+Delete()

ADMIN_EMP_DETAILS

+String : Emp_id
+String : Emp_name
+String : Emp_Gender
+String : Emp_Ph_no
+String : Emp_Email
+String : Emp_Address
+String : Emp City

+Update()
+Delete()

APPLY_LEAVE

+String : Leave_id
+String : Emp_id
+String : Cat_id
+String : Date_Start
+String : no of days
+String : Status

4.5 Er diagram

ER DIAGRAM

Emp_Password

Emp_i

Emp_Ph_n

ADMIN_EMP_DETAILS

Emp_name

Emp_Email

Emp_Gender

Emp_Addres

Status

Leave_id

Emp_id

access

retrieves

Cat_i

LEAVES_DETAILS

CATEGORY_DETAILS

Date Star

access

Cat_name

Cat_id

Date_End

Applied Date

Stored in

access

Emp_id

Applied Date

Cat_id

Status

APPLY_LEAVE

Leave_id

Date_End

Date_Start

4.6 Sequence diagram

A sequence diagram is a type of interaction diagram because it describes how—and in what order—a group of objects works together. These diagrams are used by software developers and business professionals to understand requirements for a new system or to document an existing process. Sequence diagrams are sometimes known as event diagrams or event scenarios.
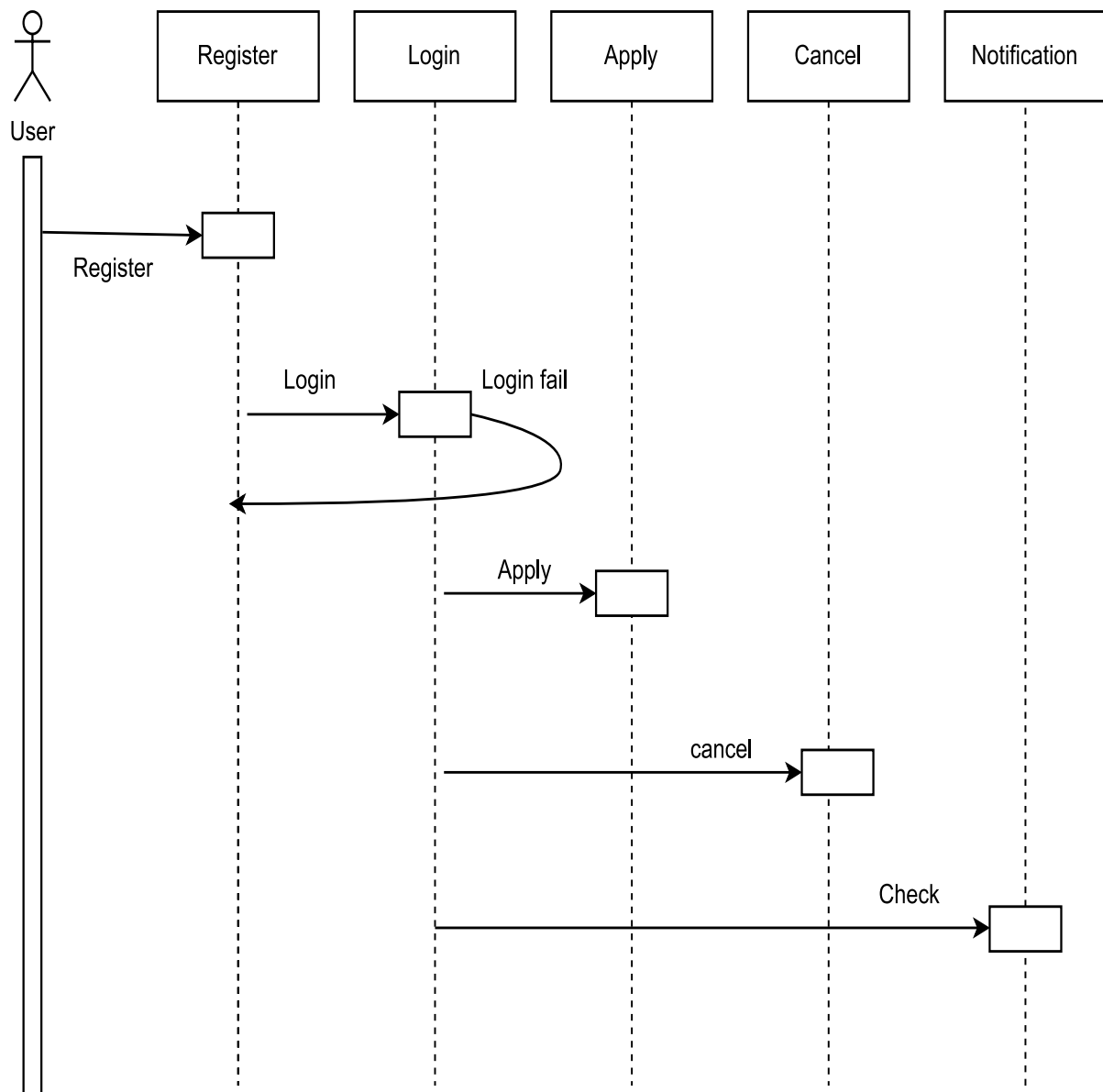
4.7 Activity diagram

Activity diagrams can be used for illustrating the sequence of project tasks. These diagrams can be created with a minimum effort and gives you a clear understanding of interdependent tasks.

```
                    (start)
                      |
                      v
                  +--------+
                  | Login  |
                  +--------+
                      |
                      v
            No     < Check if >
  +-------------+    <  valid  >
  | Invalid id &| <---
  |  password   |
  +-------------+       | Yes
                        v
  +----------+----------+----------+----------+
  |          |          |          |          |
  v          v          v          v
+--------+ +----------+ +--------+ +--------+
| View   | | Approve  | | Put    | | Manage |
| leave  | | or       | | leave  | | feedbck|
| applic.| | disapprove| |in pending| feedback|
+--------+ | leave    | +--------+ +--------+
           +----------+
```
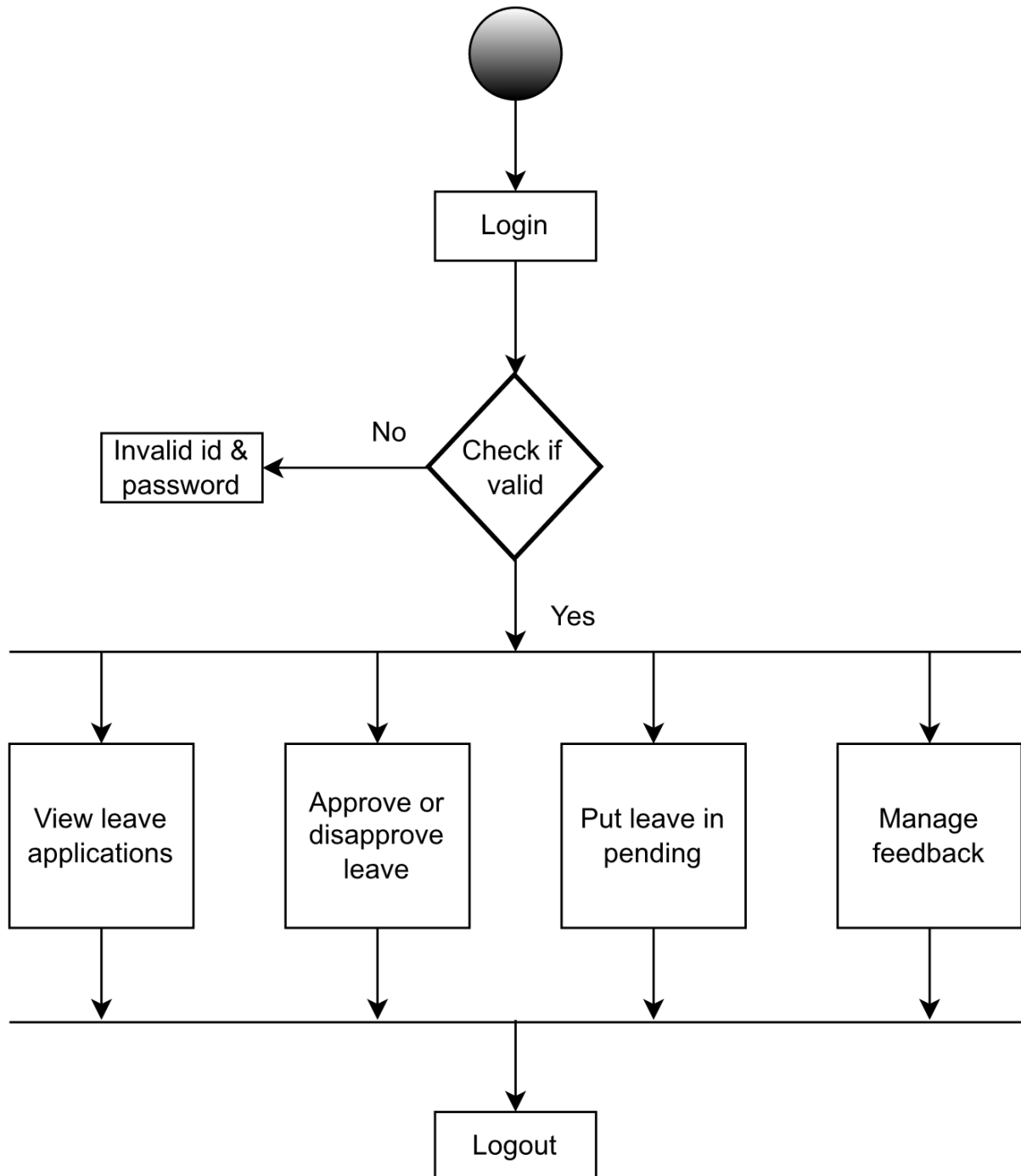
4.8 Use case diagram

Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. Hence, when a system is analysed to gather its functionalities, use cases are prepared and actors are identified. When the initial task is complete, use case diagrams are modelled to present the outside view.
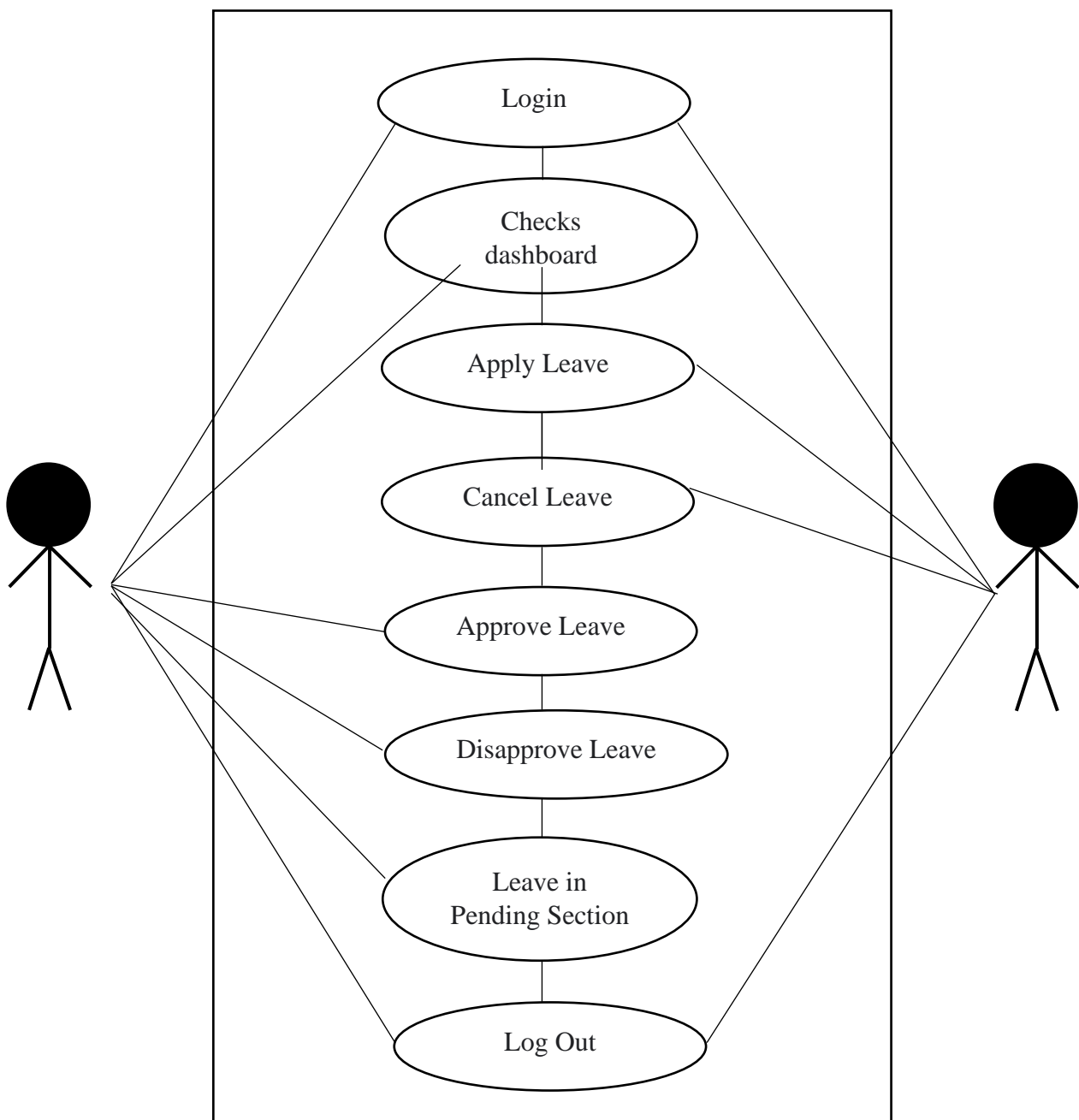
4.9 Data modeling

Data Modeling is the formalization and documentation of existing process and events that occurs during software design and development. Data modeling techniques and tools capture and translate complex system design into easily understood representation of the data flows and processes, creating blueprint for construction and re-engineering

Student detail table

Field	Data type	Constrains	Description
Employed ID	int	Primary key	Specifies id
Employee name	Varchar (100)	Not null	Specifies Employee name
gender	Varchar (100)	Not null	Specifies gender
phone	Varchar (100)	Not null	Specifies phone
Address	Varchar (100)	Not null	Specifies address
Mobile Number	Varchar (100)	Not null	Specifies mobile number

Category detail table

Field	Data type	Constrains	Description
Category ID	Int	Not null	Specifies category Id
Category name	Varchar (100)	Not null	Specifies category name
Max days	Int	Not null	Specifies max days

Apply leave

Field	Data type	Constrains	Description
Application number	int	Not null	Specifies application number
Employee id	int	Primary key	Specifies employee id
Cat Id	int	Not null	Specifies category id
Max days	Varchar (100)	Not null	Specifies max days
Starting date	Date	Not null	Specifies starting date
Number of days	Varchar (100)	Not null	Specifies status

Leave List

Field	Data type	Constrains	Description
Application number	int	Primary key	Specifies application number
Employee id	int	Not null	Specifies employee id
Cat Id	int	Not null	Specifies category id
Max days	Varchar (100)	Not null	Specifies max days
Starting date	Date	Not null	Specifies starting date
Number of days	Varchar (100)	Not null	Specifies no of days
Status	Varchar (100)	Not null	Specifies status

4.10 Modules

The system after careful analysis has been identified to be presented with the following modules and roles. The modules involve are:

EMPLOYEE MODULE:

- ➢ Admin or staff can enter Employees details.
- ➢ Employees details can be edited or deleted.

CATEGORY MODULE:

- ➢ Admin or staff can enter categories details.
- ➢ Categories details can be edited or deleted.
- ➢ They can set max days for category.

APPLY LEAVE MODULE:

- ➢ Staff apply for leave using employee detail and category detail.
- ➢ They can maintain leave details.
- ➢ Admin can search the leave details.

RESULT MODULE:

- ➢ Displays the leave detail.
- ➢ Admin can generate the result on leave.

SYSTEM DEVELOPMENT

5.1 Screenshots

LOGIN PAGE

An admin gets logged in this page. This page is mainly created for the admin so that the he/she can get logged into it. Here only the admin can interact with this page.
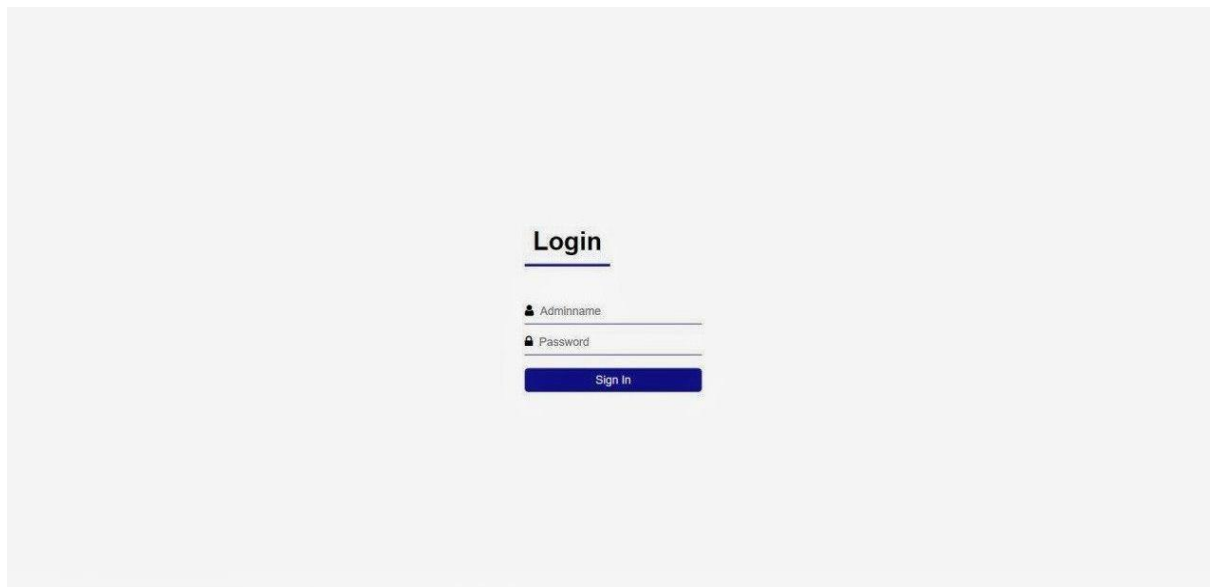
EMPLOYEE DETAILS PAGE

In this page admin or staff will enter the detail of employee and save it. Here he/she can delete, update, search the employee details which is stored in the database.

CATEGORY DETAILS PAGE

Here the admin/staff stores the information about the category which is used to take leave

Admin can delete, update and search the category details when the needed.
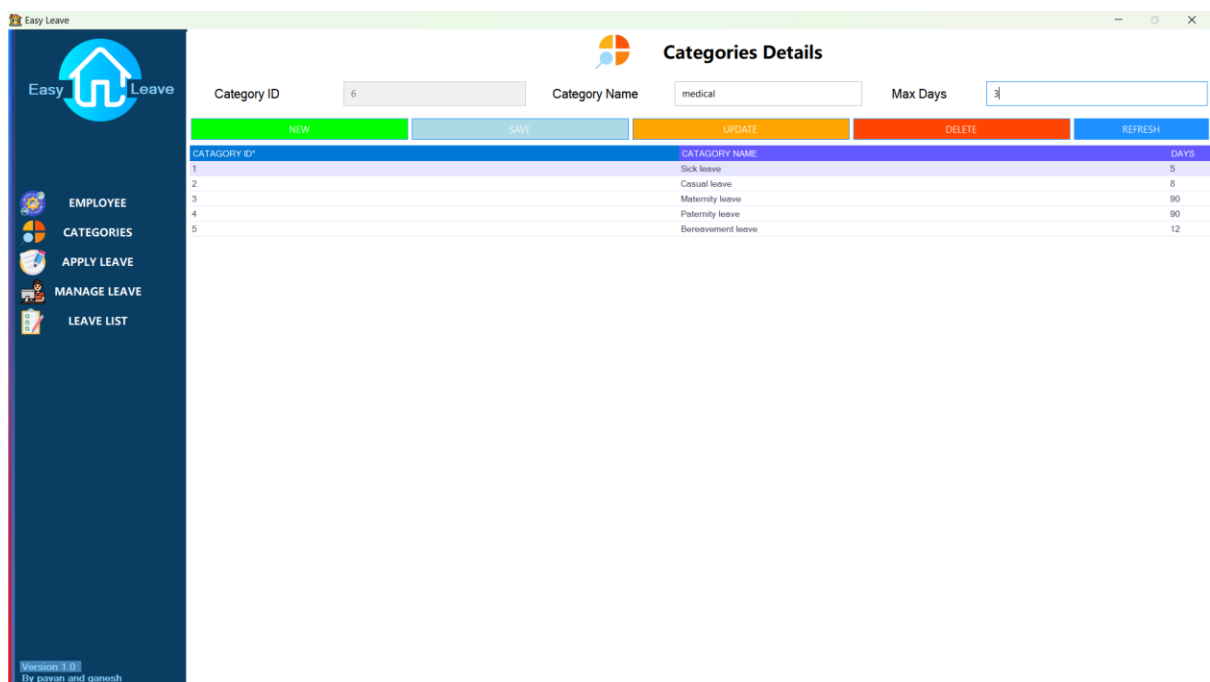
APPLY LEAVE PAGE

Leave is applied in this page using employee detail and category detail.

MANAGE LEAVE PAGE

Here the leave is approved or rejected by manager.

RESULT PAGE

In this page, admin generate the employee leave

results.

TESTING

6.1 Testing Phase

Testing is an empirical investigation conducted to provide stakeholders with information about the quality of the product or services under test, with respect to the context in which it is intended to operate. This includes, but is not limited to; the process of executing a program software with the intent of finding software program/ application/ product meets the business and technical requirements that guided its design and development, so that it works as expected and can be implemented with the same characteristics.

A primary purpose for testing is to detect software failures sop that defects may be uncovered and corrected. This is a non-trivial pursuit Testing cannot establish that a product functions properly under all conditions but can only establish that it does not function properly under specific conditions. The scope of software testing often includes examination of code as well as execution of that code in various environments and conditions well as examining the aspects of code does it do what it is supposed to do and do what it needs to do In the current culture of software development, a testing organization may be separate from the development team. There are various roles for testing team members. Information derived from software testing may be used to correct the process by which software is developed

Defects and failures: Not all software defects are caused by coding errors. One common source of expensive defects is caused by requirements gaps, eg, unrecognized Requirements, that results in errors of omission by the program designer, a common source of requirements gap in non-functional requirements such as testability, scalability. maintainability, usability, performance and security

Software faults occur through the following process. A programmer makes an error (mistakes), which results in a defect (fault, bug) in the website source code If this defect is executed, in certain situations the system will produce wrong results, causing a failure. Not all defects will necessarily result in failures. For example, defects in dead code will never result in failures. A defect can turn into a failure when the environment is changed. Examples of these changes in environment include the software being run on a new hardware platform, alterations in source data or interacting with different software. A single defect may result in a wide range of failure symptoms.

Compatibility

A frequent cause of software failure is compatibility with another software or increasingly, web browser version. In the case of lack of backward compatibility, this can occur (for example) because the programmers have only considered coding their programs for, or testing the software upon, "the latest version of this-or-that operating system. The unintended consequence of this fact is that: their latest work might not be fully compatible with earlier mixtures of software/hardware, or it might not be fully compatible with another important operating system In any case, these differences, whatever they might be, may have resulted in (unintended....) software failures as witnessed by some significant population of computers users.

Input Combinations and Preconditions

A very fundamental problem with software testing is that testing under all combinations of inputs and preconditions (initial state) is not feasible, even with a simple product. This means that the number of defects in a software product can be very large and defects that occur infrequently are difficult to find in testing. More significantly, non-functional dimensions of quality (how it supposed to be versus what it is supposed to do) for example, usability, scalability, performance, compatibility, reliability can be highly subjective; something that constitutes sufficient value to one person may be intolerable to another.

Static vs. Dynamic Testing

There are many approaches to software testing. Reviews, walkthroughs inspections are considered as static testing, whereas actually executing programmed code with a given set of test cases is referred to as dynamic testing. The former can be, (and unfortunately in practice often is....) omitted, whereas the latter takes place when programs begin to be used for the first time which is normally considered the beginning of the testing stage. This may actually begin before the program is 100% complete in order to test particular sections of code (modules or discrete functions) for example, Spreadsheet programs are, by their very nature, tested to a large extent "on the fly during the build process as the result of some calculation or text manipulation is shown interactively after each formula is entered.

1.Unit Testing

This is the smallest testable unit of a computer system and is normally tested using the white Sex testing The author of the programs usually carries out unit tests.

2. Integration Testing

Integration testing the different units of the system are integrated together to form the complete system and this type of testing checks the system as whole to ensure that it is doing what as supposed to do. The testing of an integrated system can be carried out top-down. bottom-up, or big bang In this type of testing some parts will be tested with white box testing and some with black box testing techniques. This type of testing plays very important role in increasing the systems productivity. We have checked our system by using the integration testing techniques.

3.System Testing

A part from testing the system to validate the functionality of software against the requirements, it is also necessary to test the non-functional aspect of the system Some examples of non-functional tools include test to check performance, data security, usability/user friendliness, volume, load/stress that we have used in our project to test the various modules.

System Testing Consists of the Following Steps:

- Programs(s)testing
- String testing
- System testing
- System documentation
- User acceptance testing.

4. Field Testing

This is a special type of testing that may be very important in some projects. Here the system is tested in the actual operational surroundings. The interfaces with other systems and the real world are checked. This type of testing is very rarely used. So far our project is concerned we haven't tested our project using the field testing.

5. Acceptance Testing

After the developer has completed all rounds of testing and he is satisfied with the system. then the user takes over and re-tests the system from has point of view to judge whether it is acceptable according to some previously identified criteria. This is almost always a tricky situation in the project because of the inherent conflict between the developer and the user. In this project, it is the job of the book stores to check the system that whether the made system fulfils not.

Why System Testing?

Testing is vital to the success of the system. System testing makes logical assumptions that if all the parts of the system are correct, the goal will be successfully achieved inadequate testing results in two types of problems.

1. The time lag between the cause and the appearance of the problem. 2. The of system errors on the files and records within the system.

6.2 Activity Network for System Testing

The test plan entails the following activities;

      1. Prepare test plan

      2. Specify conditions for user acceptance testing.

      3. Prepare test data for program testing

      4. Prepare test data for transaction path testing

      5. Plan user training

      6. Compile/assembles programs.

      7. Prepare performance aids.

      8. Prepare operational documents.

Prepare Test Plan

A workable test plan must be prepared in accordance with established design specifications. It includes the following items:

- Outputs expected from the system.
- Criteria for evaluating outputs.
- A volume of test data.
- Procedure for using test data.
- Personnel and training requirements.

Specify Conditions for User Acceptance Testing

Planning for user acceptance testing calls for the analyst and the user to agree on conditions for the test.

Prepare Test Data for Program Testing

As each program is coded, test data are prepared and documents to ensure that all aspects of the program are properly tested.

Prepare Test Data for Transaction Path Testing

This activity develops the data required for testing every condition and transactions to be introduced into the system. The path of each transaction from origin to destination is carefully tested reliable results.

Plan User Training

User training is designed to prepare the user for testing and converting the system, user involvement and training take place parallel with programming for three reasons:

- The system group has time available to spend on training while the programs are being written.
- Initiating a user-training program gives the system group a clearer image of the users interest in the new system.
- A trained user participates more effectively in system, testing.

The training plan is followed by preparation of the user training manual and other text materials.

Compile /Assemble Programs

All programs have to be compiled/assembled for testing

Prepare Operational Documents

During the test plan stage, all operational documents are finalized including copies of the operational formats required by the candidate system.

Systems Testing

The user to ensure that the system functions, as the user actually wanted performs this testing with prototyping techniques, this stage becomes very much a formality to check the accuracy and completeness of processing. The screen layouts and output should already have been jested during the prototyping phase.

An error in the program code can remain undetected indefinitely. To prevent this from happening the code was tested at various levels. To successfully test a system, cach condition and combinations of condition had to be tested. Each program was tested and linked to other programs. This unit of program is tested and linked to other units and so on until the complete system has been tested.

The purpose of testing is to ensure that each program is fully tested. To do so a test plan had to be created. The test plan consists of a number of test runs such as the valid paths through the code, and the exception and error handling paths. For each test run there is a list of conditions tested, the test data used and the result expected. The test plan was then reviewed to check that each path through the code is tested correctly. It is the responsibility of the programmer to collect the data that will produce the required test condition.

6.3 Verification and Validation (V&V)

The objectives of verification, validity activities are to assess and improve the quality of the work products generated during development and modification of the software. Quality depends upon the various attributes like correctness, completeness, consistency, reliability, usefulness, usability, efficiency and conformance to standards.

The terms verification and validation are used by synonymously. These are defined as under.

- Verification: "are we building the product, right?"
- Validation: "are we building right product?""
- Verification activities include proving, testing, and reviews.

Validation is the process of evaluating software at the end of the software development to ensure compliance with the software requirements. Testing is a common method of validation. Clearly, for high reliability we need to perform both activities together, they are often called V&V activities.

The major V&V activities for software development are inspection, reviews, and testing (both static and dynamic) the V&V plan identifies the different V&V task for the different phases and specifies how these tasks contribute to the project V&V gas the methods t be used for these V&V activities, the responsibilities and milestones for each of these activities, inputs and outputs for each V&V tasks, and criteria for evaluating the outputs are as specified.

The two major V&V approaches are testing and inspections Testing is an activity that can be generally performed only on code. It is an important activity and is discussed in detail in a later chapter. Inspection is a more general activity that can be applied to any product, including code. Many of the V&V tasks are such that for them, an inspection type of activity is the only possible way to perform the tasks (e.g. trace ability and document evaluation). Due to this, inspections play a significant role in verification.

SYSTEM IMPLEMENTATION

7.1 Post Implementation Maintenance and Review

As we know, creating software is one among the implementation of the created software is another. The process of implementing software is much difficult as compared to the task of creating the project. First, we have to implement the software on a small scale. Before we think in terms of implementing the software on a large basis, we must consider the Hardware requirements.

Whatever we develop software or project certain hardware and software is being used by the programmer for developing the project. The hardware and software to be used by the programmer for developing the project should be such that it would result in which the project has been created by the programmer. The hardware should be such that cost constraints of the client should also be taken into account without affecting the performance.

7.2 Hardware Evaluation Factors

When we evaluate computer hardware, we should first investigate specific physical and performance characteristics for each hardware component to be acquired This specific question must be answered concerning many important factors. These hardware evaluation factors questions are summarized in the below figure

Notice that there is much more to evaluating hardware than determining the faster and cheaper computing device. For eg the question of possible obsolescence must be addressed by making a technology evaluation. The factor of ergonomics is also very important. Ergonomics is the science and technology that tries to ensure those computers and other technologies are "user-friendly", that is safe, comfortable and easy to use Connectivity is another important evolution factor, area telecommunication networks.

Hardware Evaluation Factors

1. Performance

2. Cost

3. Reliability

4. Availability

5. Compatibility

6. Modularity

7. Technology

8. Ergonomics

9. Environmental requirements

10. Software

11.Support

7.3 Software Evaluation Factors

Software can be evaluated according to many factors similar to the hardware evaluation. Thus, the factors of performance, cost, reliability, compatibility, modularity, technology, ergonomics, and support should be used to evaluate proposed software acquisitions. In addition, however, the software evaluation factors are summarized in below figure. For eg some software packages require too much memory capacity and are notoriously slow, hard to use or documented. They are not a good selection for most end users, even if offered at attractive prices.

Software Evaluation Factors:

- **Efficiency:** is the software a well-written system or computer instructions that does not use much memory capacity or CPU time.
- **Flexibility:** can it handle its processing assignments easily without major modifications?
- **Security:** does it provide control procedure for errors, malfunctions and improper use?
- **Language:** do our computer programmers and users write it in a programming language   that is used?
- **Documentation:** is the s/w well documented? Does it include helpful user instruction?
- **Hardware:** does existing hardware have the features required to best use this software?
- **Other Characteristics:** of hardware such as its performance, what about the cost, how much is reliable and etc.

7.4 Conversion and Training

An important aspect of is to make sure that the new design is implemented to establish standards. The term implementation has different meanings, ranging from the conversion of a

basic software to a complete replacement of a converting a new or revise system into an operational one. Conversion is one aspect of implementation. Conversion means changing from one system to another. The objectives is to put the system into operation while holding cost, risks, and personnel irritation to a minimum it involves creating computer-compatible files, training the operation staff, and installing terminal and hardware. A critical aspect of conversion is not disrupting the functioning of the organization.

When a new system is used over and old, existing and running one, there are always compatibility errors. These errors are caused because of the lack of equipment or personnel to work the new system. Running any specified system at an organization does require some or other hardware or, in this case software requirement as well.

Conversion is one aspect of implementation review & software maintenance. There are three types of implementations

1. Implementation of a computer system to replace a manual system. The problems encountered are converting files, training users, creating accurate files and verifying printouts for integrity

2. Implementation of a new computer system to replace an existing one. This is usually a difficult conversion. If not properly planned there can be many problems. Some large computer systems have taken as long as year to convert.

3. Implementation of a modified software to replace an existing one. Using the same computer. This type of conversion is relatively easy to handle, provided there are no major changes in the files.

7.5 Training needs

Training needs refer to the gaining of knowledge required for running the system. First of all the system is a computer based system therefore the person should have good knowledge about computer and it's on the computer. For the better usage and working of the WEBSITE the organization should appoint a person who has good knowledge of all the required software. The organization gets a person trained through different institute 's person in the market. The training should be as per the above requirements.

COST ESTIMATION OF THE PROJECT

Cost Estimation of The Project

Cost in a project is due to the requirements for software, hardware, and human resources. Hardware resources are computer time, terminal time and memory required for the project. Software resources include the tools and compliers needed during development. The bulk cost of software development is due to human resources needed. Cost estimates are determined in terms of persons per months (PM).

Total number of persons involved in this project:

1. Administrator
2. Senior programmer
3. Junior programmer
4. User of a software

Since this project will complete in 4 months

- **COST ESTIMATE:** (Salary of Project Manager + Salary of Senior Programmer + 2*Salary of Junior programmer) *2

  ➢ **NOTE:** This software is made by me under the guidance of my guide, so the estimation is shown here is if company prepares the software.

8.1 Gantt & Pert Chart

GANTT CHART

Gantt charts mainly used to allocate resources to activities. The resources allocated to activities include staff, hardware, and software. A Gantt chart is named after its developer Henry Gantt. It is useful for resource planning. A Gantt chart is special type of bar represents an activity. The bars are drawn along a timeline. The length of each bar is proportional to the duration of the time planned for the corresponding activity

Gantt chart is a project scheduling technique. Progress can be represented easily in a Gantt chart, by colouring each milestone when completed.

GANTT CHART

|	December				January				February		
	1week	2week	3week	4week	1week	2week	3week	4week	1week	2week	3week
Front End											
Create use case	█	█									
Select design team	█	█									
design front end			█	█	█						
Integration &testing						█	█	█			
Launch									█	█	
Build front end											█
Beta sprinters											
Design build serving		█	█	█	█	█	█	█			
Platform											
Technical design	█	█									
Assemble development team	█	█									
Build plat form (Phase1)		█	█	█							
Build plat form (Phase2)					█	█	█				
Integration &testing									█	█	
Launch											█
Context load & QA (for testing)			█	█							
Context load & QA (for front end)					█	█					
Context load & QA (for beta sprinters)							█	█			

PERT CHART

PERT (Project Evaluation and Review Technique) charts consist of network of boxes and arrows. The boxes represent activities and the arrows represent task dependencies.

PERT chart represents the statistical variations in the project estimates assuming a normal distribution. Thus, in a PERT chart instead of making a single estimate for each task, pessimistic, likely, and optimistic estimates are also made. The boxes of PERT charts are usually annotated with the pessimistic, likely, and optimistic estimates for every task. Since all possible completion time between the minimum and maximum durations for every task have to be considered, there are many critical paths, depending on the permutations of the estimates for each task. This makes critical path analysis in PERT charts very complex. A critical path in PERT chart is shown by using thicker arrows. The PERT chart representation of the company's problem of Figure A. is shown in Figure B.

PERT Charts are a more sophisticated form of activity chart. In activity diagrams only the estimated task durations are represented. Since the actual durations might vary from the estimated durations, the utility of the activity diagrams is limited.

PERT CHART
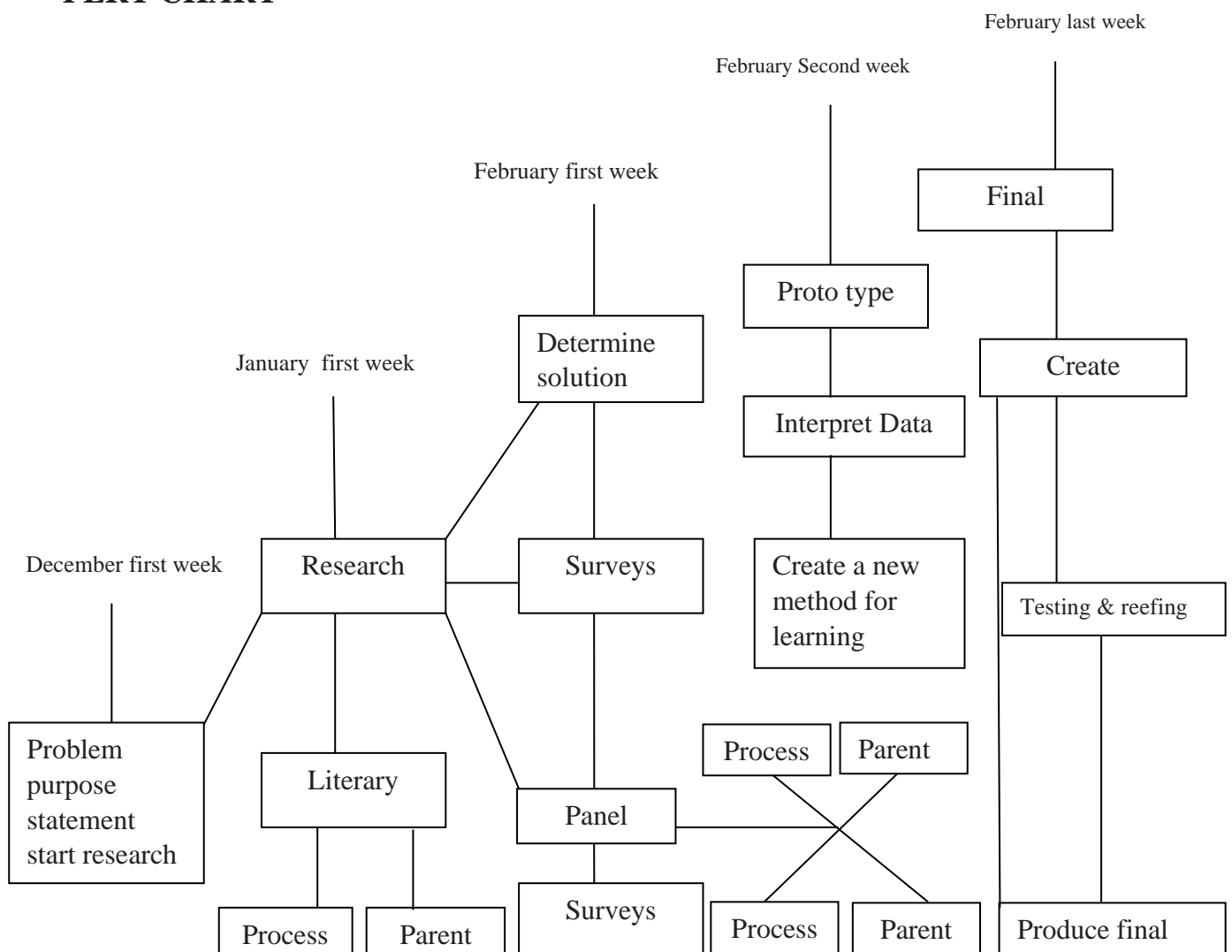
SECURITY AND VALIDATION CHECK

9.1 Security and validation check

9.2 Software's vulnerability to attack

Software development is not yet a science or rigorous discipline, and the development process by and large is not controlled to minimize the vulnerabilities that attackers exploit

The security of software is threatened at various points throughout its life cycle, both by inadvertent and international choices and actions taken by "insiders"- individuals closely affiliated with the organization that is producing, deploying, operating, or maintaining the software, and thus trusted by the organization. The software security can be threatened.

- **During its development**: A developer may corrupt the software intentionally or unintentionally or unintentionally-in ways that will compromise the software's dependability and trustworthiness when it is operational.

- **During its deployment (distribution and installation)**: If those responsible for distributing the software fail to tamperproof the software before shipping or uploading, or transmit it over easily intentional or unintentional corruption. Similarly, if the software's installer fails to "lock down" the host platform, or configures the software insecurely, the software is left vulnerable to access by attackers.

- **During its operation**: once cots and open-source software has gone operational, vulnerabilities may be discovered and publicized; unless security patches and updates are applied and newer supported versions (from which the root causes of vulnerabilities have been eliminated) are adopted, such software will become increasingly vulnerable. Non _commercial software and open-source software (OSS) may also be vulnerable, especially as it may manifest untrustworthy behaviours over time due to changes in its environment that stress the software in ways that were not anticipated and simulated during its testing.

- **During its sustainment:** if those responsible for addressing discovered vulnerabilities in released software fail to issue patches or updates in a timely manner, or fail to seek out and eliminate the root causes of the vulnerable vulnerabilities to prevent their perpetuation in future releases of the software, the software will become increasingly vulnerable to threats over time. Also, the software's maintainer may prove to be a malicious insider, and may embed malicious code, exploitable flaws, etc., in updated versions of the code.

The challenges of building secure software

1. **Dependability:** dependable software executes predictably and operates correctly under all conditions, including hostile conditions, including when the software comes under attack or runs on a malicious host.

2. **Trustworthiness:** Trustworthy software contains few if any vulnerabilities or weaknesses that can be intentionally exploited to subvert or sabotage the software's dependability. In addition, to be consider Trustworthy. The software must contain no malicious logic that causes it to behave in a malicious manner.

3. **Survivability (also referred to as "Resilience"):** Survivable-or resilient- software is software is software that is resilient enough to (1) either resist (i.e., protect itself against) or tolerate (i.e., continue operating dependably in spite of) most known attacks plus as many novel attacks as possible, and (2) recover as quickly as possible, and with as little damage as possible, from those attacks that it can neither resist nor tolerate.

9.3 Software assurance

The main objective of software assurance is to ensure that the processes, procedures, and products used to produced and sustain the software conform to all requirements and standards specified to govern those processes, procedures and products. Software security and secure software are often discussed in the context of software assurance. Software assurance in its broader sense refers to the assurance of any required property of software.

An increasingly agreed-upon approach for assuring the security of software is the software security assurance case, which is intended to provide justifiable confidence that the software under consideration (1) is free of vulnerabilities;(2)functions in the intended manner", and this "intended manner "does not compromise the security or any other required properties of the software, its environment, or the information it handles; and (3) can be trusted to continue operating dependably under all anticipated circumstances, including anomalous and hostile environmental and utilization circumstances-which means that those who build the software need to anticipate such circumstances and design and implement the software to be able to handle them gracefully. Such circumstances include

- The presence of unintentional faults in the software and its environment
- the exposure of the operational software to accidental events that threaten its security

- the exposure of the software to intentional choices or actions that threaten its security during its development, development, operation, or sustainment

Software is more likely to be assuredly secure when security is a key factor in following aspects of its development and deployment.

Development principles and practices: the practices used to develop the software and the principles that governed its development are expressly intended to encourage and support the consideration and evaluation of security in every phase of the software's development life cycle. Some secure development principles and practices for software are suggested later in this article.

Development tools: the programming languages(s), libraries, and development tools used to design and implement software are evaluated and selected for their ability to avoid security vulnerabilities and to support secure development practices and principles.

Testing practices and tools: the software is expressly tested to verify its security, using tools that assist in such testing.

Acquired components: commercial off-the-shelf (COST) and OSS components are evaluated to determine whether they contain vulnerabilities. And if so whether the vulnerabilities can be remediated through integration to minimize the risk they pose to the software system.

Deployment configuration: the installation configuration of the software minimizes the exposure of any residual vulnerability it contains.
Execution environment: protections are provided by the execution environment that can be leveraged to protect the higher level software that operates in that environment

Practitioner knowledge: the software analyst, designers, developers, testers, and maintainers are provided with the necessary information (example through training and education) to give them sufficient security awareness and knowledge to understand, appreciate, and effectively adopt the principles and practices that will enable them.

FUTURE SCOPE OF SOFTWARE

Scope of The Project

A leave management system is a software, designed to help company to manage their day-to-day leave.

The project made here is just to ensure that the leave management to store the records of employee, category info about leave details and generate the results.

Currently the system works for the staff and administrator. staff can store the information. It provides details directly to the admin or the faculties when they needed. And admin takes the decision on the leave.

.

Users of the System

- Manager
- staff

Limitations

- Staff has to store the info manually
- It was less user - friendly.
- Admin or main staff can operate the software.

10.1 Conclusion

- This LEAVE MANAGEMENT SYSTEM helps the admin to view and manage the leaves applied by their users. A leave management system helps in recording, managing, and tracking employees' time-off requests. Its main objective is to handle employees' leave requests impartially while ensuring that the employees' absence from work doesn't adversely impact the business.

- This software helps the admin leading to saving time.

- This management system provides the approval, disapproval, pending and notifying functions which makes the work of admin easier.

10.2 BIBLOGRAPHY

Name	Author	Publication
Design Pattern	Erich Gamma	1994
Clean code	Robert martin	2002
Rapid Development	Steve McConnell	1996

Websites referred are:

- **www.slideshare.com**
- **www.freeproject.com**
- **www.nevonproject.com**
- **www.javatpoint.com**