Project Name **: Zero Queue**

Submitted By **: Rahina Banu O**

Program Code **: MCA**

Enrollment Number **: 2001636705**

Regional Center Code **: 14 Cochin**

Course code **: MCSP 060**

Mobile Number **: 9526663275, 9605824223**

Email id **: rahinabanu797@gmail.com**

INDIRA GANDHI NATIONAL OPEN UNIVERSITY

MCSP - 060

ZERO QUEUE

by

Rahina Banu O

Enrolment No: 2001636705

Under Guidance of

Ashik NC

Submitted to the School of Computer and Information Sciences, IGNOU

in partial fulfilment of the requirements

for the award of the degree

Master of Computer Applications (MCA)

2024

Indira Gandhi National Open University

Maidan Garhi

New Delhi – 110068.

ZeroQueue

TABLE OF CONTENTS

INTRODUCTION

Zero Queuing system automates the manual queues. The system provides an android based application where a user can create or join a virtual queue. The application provides an effortless and time saving way of managing queues using which organizations can help their customers or clients without making them wait in long queues also users are in turn benefitted by queuing at a place without wasting their precious time and efforts. Many organizations are encountering colossal challenges and difficulties during the pandemic period. Many people around the globe are not even visiting the organizations even for their needs thinking that there is a huge possibility of getting the spread disease during the waiting time in the queue. So, in order to drive out the fear from the people and to provide a minimum staying time in the organizations premises during the visit, research work proposes a Zero Queue Management System (ZQR4S). This system also provides the people to book an appointment with the organizations based on availability of the schedule. User can book for each department separately by using this application. It also supports the users to cancel the appointment in case of change of plans. User can also make use of waiting list option available in the mobile application to enrol their name in waiting list in order to intimate them regarding any cancellation of appointment.

OBJECTIVES

THE MAIN OBJECTIVES OF THE PROPOSED SYSTEM ARE

- In comparison to the present system the proposed system will be less time consuming and more efficient.
- Analysis will be very easy in proposed system as it is automated.
- Result will be very precise and accurate.
- The data are stored in a database and can be back up for future use.

SYSTEM ANALYSIS

System analysis in software engineering is, therefore, the activities that comprise software engineering as a process in the production of software. It is the software process. This process has 4 main activities. They are:

- Software specification
- Software design and implementation
- Software validation
- Software evolution

As we can see, these activities are similar to those within systems analysis and the design of software. Depending on the methodology used, the activities can be arranged differently. They are arranged sequentially, for example, in the well-known Waterfall Model, while in the Incremental Development model they are inter-related.

IDENTIFICATION OF NEED

Typically, a project is proposed by an individual who identifies a project- worthy need or opportunity. The Identification phase of the IT Project Management Framework involves evaluating and deciding if a proposed project should be undertaken, based on the studying of factors like costs, benefits, risks, and etc.

PRELIMINARY INVESTIGATION

Preliminary Investigation basically refers to the collection of information that guides the management of an organization to evaluate the merits and demerits of the project request and make an informed judgment about the feasibility of the proposed system. This sort of investigation provides us with a through picture of the kind of software and hardware requirements which are most feasible for the system, plus the environment in which the entire project has to be installed and made operational.

FEASIBILITY STUDY

A feasibility study is a preliminary study undertaken to determine and document a project's viability. The results of this study are used to make a decision whether to proceed with the project. If it indeed leads to a project being approved, it will - before the real work of the proposed project starts - be used to ascertain the likelihood of the project's success. It is an analysis of possible alternative solutions to a problem and a recommendation on the best alternative. It, for example, can decide whether an order processing be carried out by a new system more efficiently than the previous one. The feasibility study proposes one or more conceptual solutions to the problem set for the project. The conceptual solution gives an idea of what the new system will look like. They define what will be done on the computer and what will remain manual. It also indicates what input will be needed by the system and what outputs will be produced. These solutions should be proven feasible and a preferred solution is accepted.
The feasibility study environment enables all alternatives to be discussed and evaluated. This phase starts with an identification of the main characteristics of the required system. During this stage it is important to collect information as much as possible about the software package that might meet the specification from as many sources as possible.

Normally, the central endeavour of a feasibility study is a cost benefit analysis of various alternatives. It can be defined as a systematic comparison between the cost of carrying out a service or activity and the value of that service or activity. The main benefits are qualitative than quantitative.

A feasibility study could be used to test a new working system, which could be used because:

- The current system may no longer suit its purpose,
- Technological advancement may have rendered the current system obsolete,
- The business is expanding, allowing it to cope with extra work load,
- Customers are complaining about the speed and quality of work the business provides

Competitors are now winning a big enough market share due to an effective integration of a computerized system.

When a new project is proposed, it normally goes through feasibility assessment. Feasibility study is carried out to determine whether the proposed system is possible to develop with available resources and what should be the cost consideration.
Facts considered in the feasibility analysis were

 Technical Feasibility
 Operational Feasibility
 Economic Feasibility

Technical Feasibility

This involves questions such as whether the technology needed for the system exists, how difficult it will be to build, and whether the firm has enough experience using that technology. The assessment is based on an outline design of system requirements in terms of Input, Output, Fields, Programs, and Procedures. This can be qualified in terms of volumes of data, trends, frequency of updating etc. in order to give an introduction to the technical system.
The system requires normal configuration computer systems that are commonly available. The software requirements are C# .Net, Windows 8 or higher versions of OS. Thus proposed system is technically feasible.

Operational Feasibility

This analysis involves how it will work when it is installed and the assessment of political and managerial environment in which it is implemented. People are inherently resistant to change and computers have been known to facilitate change. The new proposed system is very much useful to the users and there for it will accept broad audience.
The proposed system offers:

- Greater user friendliness
- Better output which can be easily interpreted.
- Higher speed.
- Meets the requirements of the organizations.

Economic feasibility

This involves questions such as whether the firm can afford to build the system, whether its benefits should substantially exceed its costs, and whether the project has higher priority and profits than other projects that might use the same resources. This also includes whether the project is in the condition to fulfil all the eligibility criteria and the responsibility of both sides in case there are two parties involved in performing any project.
This study presents tangible and intangible benefits from the project by comparing the developments and operational costs. The technique of cost benefit analysis is often used as a basis for assessing economic feasibility. This system needs some more initial investment than the existing system, but it can be justifiable that it will improve the quality of service. Thus, feasibility study should centre along the following points:

- Improvement resulting over the existing method in terms of accuracy, timeliness.
- Cost comparison.
- Estimate on the life expectancy of the hardware.
- Overall objective.

PROJECT PLANNING AND SCHEDULING

For the successful completion of every project there must be detailed scheduling. The software development has different participating steps. First of all, I had done the requirement analysis phase. For this I visit different sites that offer resume writing helps, visits different websites, and I discuss with my friends and project guide. After collecting the requirements, a detailed study of preliminary investigation is done. After the analysis phase the requirements and document design are divided into modules. The document is created, which includes dataflow diagrams, ER diagrams etc.

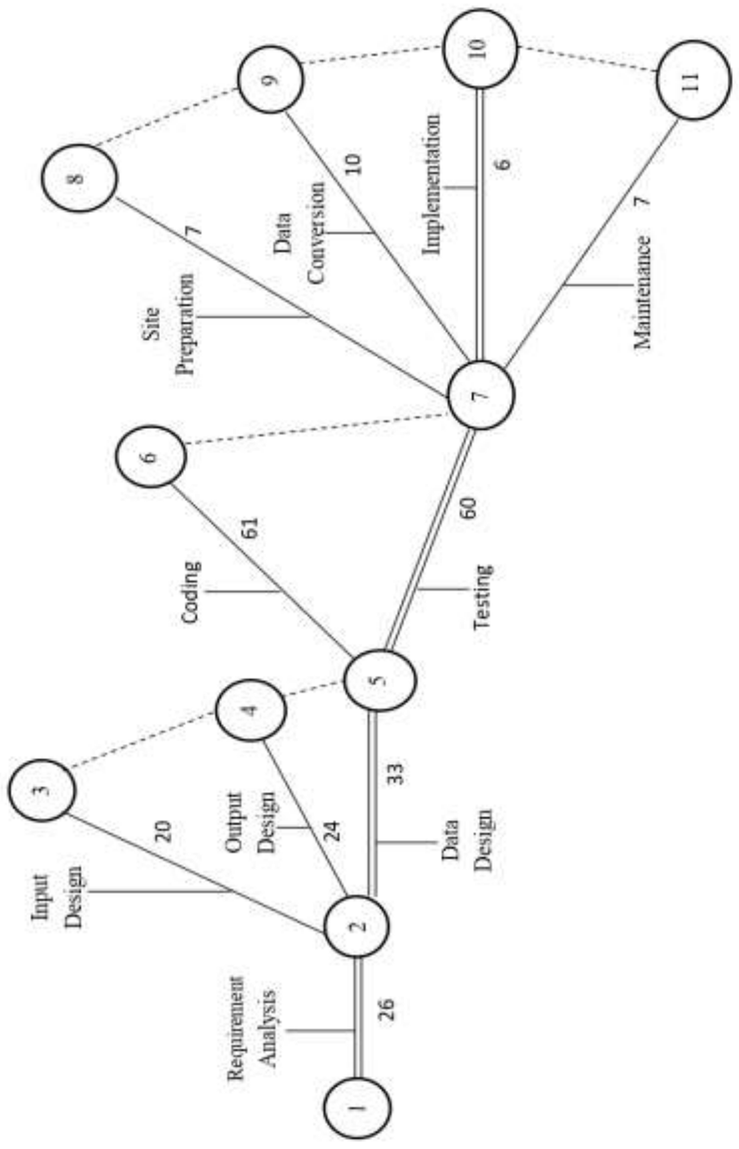
As next step the actual development of the system takes place. The design representations are translated into codes. Documentation of codes is done by providing explanations of how procedures are used. Documentation is essential to test the program and carry on maintenance once the application has been installed.

As next step testing is done. After a system has been developed it is very important to check if it fulfils the user requirements. Implementation of the system means putting up system on user’s side. Like any system there is an aging process. Therefore, the system requires periodic maintenance for software or hardware.

GANTT CHART

Month	JANUARY 1- JANUARY 31								FEBRUARY 1- FEBRUARY 28								MARCH 1- MARCH 31							
Days	4	4	4	4	4	4	4	3	4	4	4	4	4	4	4	2	4	4	4	4	4	4	4	3
Requirement analysis:	■	■	■	■	■	■	■	■	■	■	■	■					▨	▨	▨	▨				
Analysis	▨	■	■	■	■	■	■	■	▨	▨	▨	▨					▨	▨	▨	▨				
Milestone: Analysis complete	▨	▨	▨	▨			■	■	■	■	■	■					▨	▨	▨	▨				
Design:	▨	▨	▨	▨					▨	▨	▨	▨	■	■	■	■	■	▨	▨	▨				
i. Input Design	▨	▨	▨	▨					▨	▨	▨	▨		■	■	■	▨	▨	▨	▨				
ii. Output Design	▨	▨	▨	▨					▨	▨	▨	▨			■	■	■	▨	▨	▨				
iii. Data Design	▨	▨	▨	▨					▨	▨	▨	▨			■	■	■	▨	▨	▨				
Milestone: Design complete	▨	▨	▨	▨					▨	▨	▨	▨				■	■	▨	▨	▨				
Coding & Testing	▨	▨	▨	▨					▨	▨	▨	▨					■	■	■	■	■			
i. Coding	▨	▨	▨	▨					▨	▨	▨	▨					▨	■	■	■	■			
ii. Testing	▨	▨	▨	▨					▨	▨	▨	▨					▨	▨	■	■	■			
Milestone: Coding & Testing complete	▨	▨	▨	▨					▨	▨	▨	▨					▨	▨	▨	■	■			
Implementation & Maintenance:	▨	▨	▨	▨					▨	▨	▨	▨					▨	▨	▨	■	■	■	■	■
i. Site Preparation	▨	▨	▨	▨					▨	▨	▨	▨					▨	▨	▨	▨	■	■	■	
ii. Data Conversion	▨	▨	▨	▨					▨	▨	▨	▨					▨	▨	▨	▨	■	■	■	
iii. Implementation	▨	▨	▨	▨					▨	▨	▨	▨					▨	▨	▨	▨		■	■	■
iv. Maintenance	▨	▨	▨	▨					▨	▨	▨	▨					▨	▨	▨	▨			■	■
Milestone: Implementation & Maintenance complete	▨	▨	▨	▨					▨	▨	▨	▨					▨	▨	▨	▨			■	■

Pert Chart



Numbers on the edge are represented in Number of days.

SOFTWARE REQUIREMENTS SPECIFICATION

A software requirements specification (SRS) is a description of a software system to be developed, laying out functional and non-functional requirements. (Non-functional requirements impose constraints on the design or implementation such as performance engineering requirements, quality standards, or design constraints.) The specification may include a set of use cases that describe interactions the users will have with the software. The software requirements specification document enlists enough and necessary requirements that are required for the project development. To derive the requirements, we need to have clear and thorough understanding of the products to be developed or being developed. This is achieved and refined with detailed and continuous communications with the project team and customer till the completion of the software.

SYSTEM SPECIFICATION

HARDWARE SPECIFICATION

The selection of hardware is very important in the existence and proper working of any of the software. When selecting hardware, the size and capacity requirements are also important. The hardware must suit all application developments.

• Processor	:	i3 or above.
• System Bus	:	32Bit or 64Bit
• RAM	:	4 GB or Above
• HDD	:	500 GB or Above
• Monitor	:	14” LCD or Above
• Key Board	:	108 Keys
• Mouse	:	Any Type of mouse

SOFTWARE SPECIFICATION

One of the most difficult tasks is selecting software, once the system requirement is find out then we have to determine whether a particular software package fits for those system requirements. This section summarizes the application requirement.

- □ Operating System : Windows 10 Any 32 bit or 64 bit platform
- □ Front End : Python
- □ Back End : MySQL Sever
- □ IDE : Python 3.6 or above, PyCharm

ABOUT THE FRONT END

An Integrated Development Environment (IDE) (also known as Integrated Design Environment or Integrated Debugging Environment) is a software application that provides comprehensive facilities to computer programmers for software development. An IDE normally consists of:

- A source code editor
- A compiler and/or an interpreter
- Build automation tools
- A debugger

PYTHON

Python is an interpreted high-level general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured, object- oriented and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library. Guido van Rossum began working on Python in the late 1980s, as a successor to the ABC programming language, and first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000 and introduced new features, such as list comprehensions and a garbage collection system using reference counting. Python 3.0 was released in 2008 and was a major revision of the language that is not completely backward compatible and much Python 2 code does not run unmodified on Python 3. Python 2 was discontinued with version 2.7.18 in 2020. Python consistently ranks as one of the most popular programming languages. Python was conceived in the late 1980s by Guido van Rossum at Centrum Wiskunde & Informatics (CWI) in the Netherlands as a successor to ABC programming language, which was inspired by SETL, capable of exception handling and interfacing with the Amoeba operating system. Its implementation began in December 1989. Van Rossum shouldered sole responsibility for the project, as the lead developer, until 12 July 2018, when he announced his "permanent vacation" from his responsibilities as Python's Benevolent Dictator For Life, a title the Python community bestowed upon him to reflect his long-term commitment as the project's chief decision-maker.

ABOUT THE BACK END

DATABASE SERVERS

A database server is used to store data in a database. Users can access the data and manipulate it. There are many types of databases. The most popular among them is the Relational Database Management System (RDBMS).

RDBMS

RDBMS is a type of database management system that stores data in the form of related tables. Relational database are powerful because they require few assumptions about how data is related or how it will be extracted from the database. As a result, the same database can be viewed in many different ways. An important feature of relational systems is that a single database can be spread across several tables. This differs from flat-file database, in which each database is self-contained in a single table.

SQL

SQL (pronounced "ess-que-el") stands for Structured Query Language. SQL is used to communicate with a database. According to ANSI (American National Standards Institute), it is the standard language for relational database management systems. SQL statements are used to perform tasks such as update data on a database, or retrieve data from a database. Some common relational database management systems that use SQL are: Oracle, Sybase, Microsoft SQL Server, Access, Ingres, etc. Although most database systems use SQL, most of them also have their own additional proprietary extensions that are usually only used on their system. However, the standard SQL commands such as "Select", "Insert", "Update", "Delete", "Create", and "Drop" can be used to accomplish almost everything that one needs to do with a database. This tutorial will provide you with the instruction on the basics of each of these commands as well as allow you to put them to practice using the SQL Interpreter.

SOFTWARE ENGINEERING PARADIGM APPLIED

In our project I have used RAD model. Rapid application development (RAD) is an incremental software development process model; that emphasizes an extremely short development cycle. The RAD model is a “high-speed” adaptive of the linear sequential model in which rapid development is achieved by using components-based construction. If requirements are well understood and project scope is constraint, the RAD process enables a development team to create a “fully functional system “with in very short time periods. The RAD approach encompasses the following phase:

- Business modelling
- Data modelling
- Process modelling
- Application modelling
- Testing and turnover

DATA MODELS

1. DFD

A graphical representation is used to describe and analyze the movement of data through a system manual or automated including the processes, storing of data and delays in the system. Data flow diagrams are the central tool and the basis from which other components are developed.
The transformation of the data from input to output through process may be described logically and independently of the physical components associated with the system. They are termed logical data flow diagrams, showing the actual implementation and the movement of data between people, departments and workterminuss. DFD is one of the most important modeling

tools used in physical design. DFD shows the flow of data through different process in the system.

The purpose of the design is to create architecture for the evolving implementation and to establish the common tactical policies that must be used by desperate elements of the system. We begin the design process as soon as we have some reasonably completed model of the behavior of the system. It is important to avoid premature designs, wherein develop designsbefore analysis reaches closer. It is important to avoid delayed designing where in the organization crashes while trying to complete an unachievable analysis model.

Data flow diagram is quite effective, especially when the required design is unclear and the user and analyst need a notational language for communication. It is one of the most important tools used during system analysis. It is used to model the system components such as the system process, the data used by the process, any external entities that interact with the system and information flows in the system. Data flow diagrams are made up of a number of symbols, which represents system components. Data flow modeling method uses four kinds of symbols

Process

Process shows the work of the system. Each process has one or more data inputs and produce one or more data outputs. Processes are represented by circles in data flow diagrams.

Data Stores

A data store is a repository of data. Processes can enter data into a store or retrieve data from the data store. Data stores are represented by two parallel lines, connected with two horizontal lines. Which may be depicted horizontally or vertically?

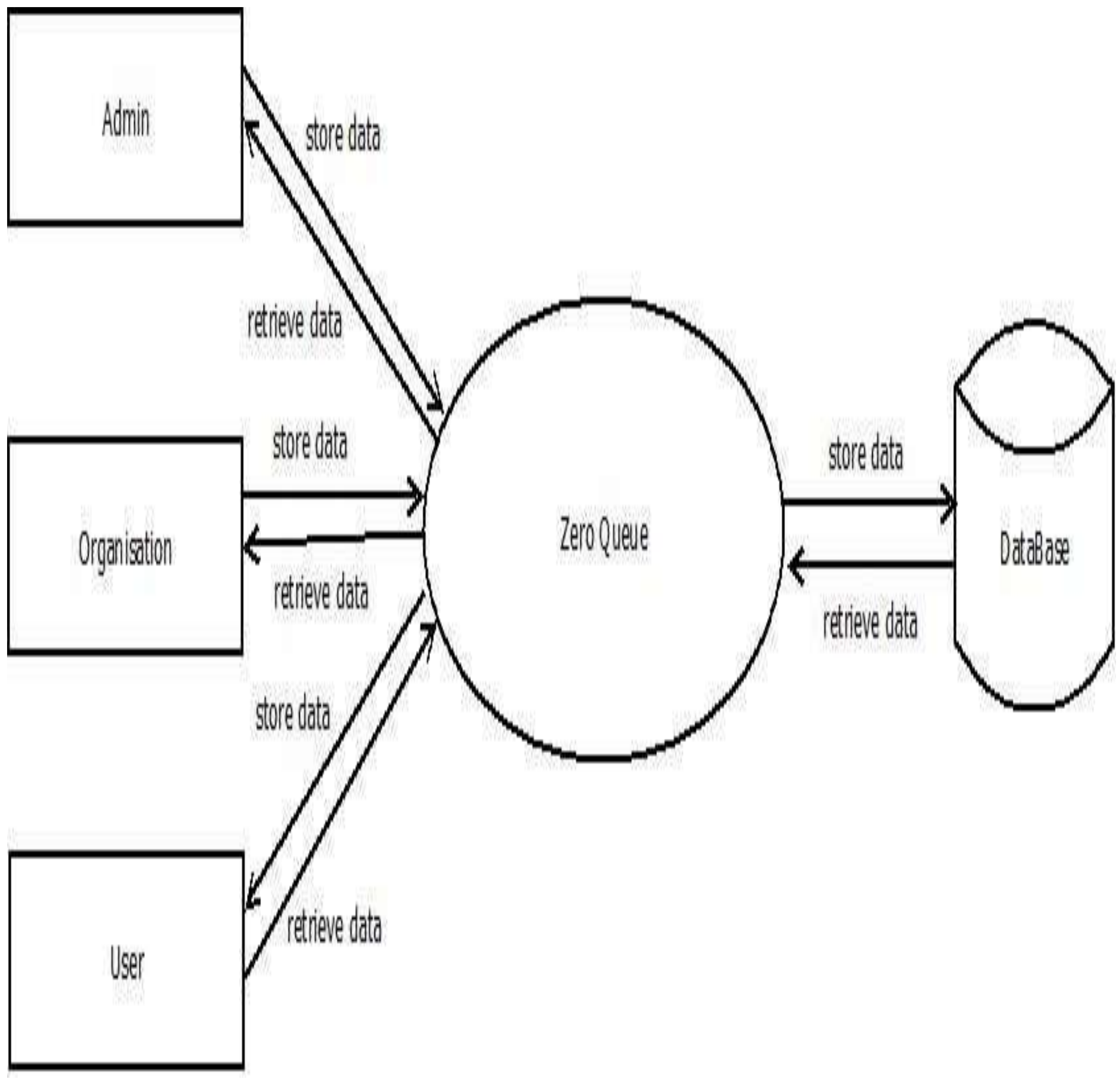
Data Flows

The arrows represent data flow. A data flow is data in motion. A data flow represents an input of data to a process or the output of the data from a process. A data flow is also used to represent the creation, reading, deletion, or updating of data in a file or database.

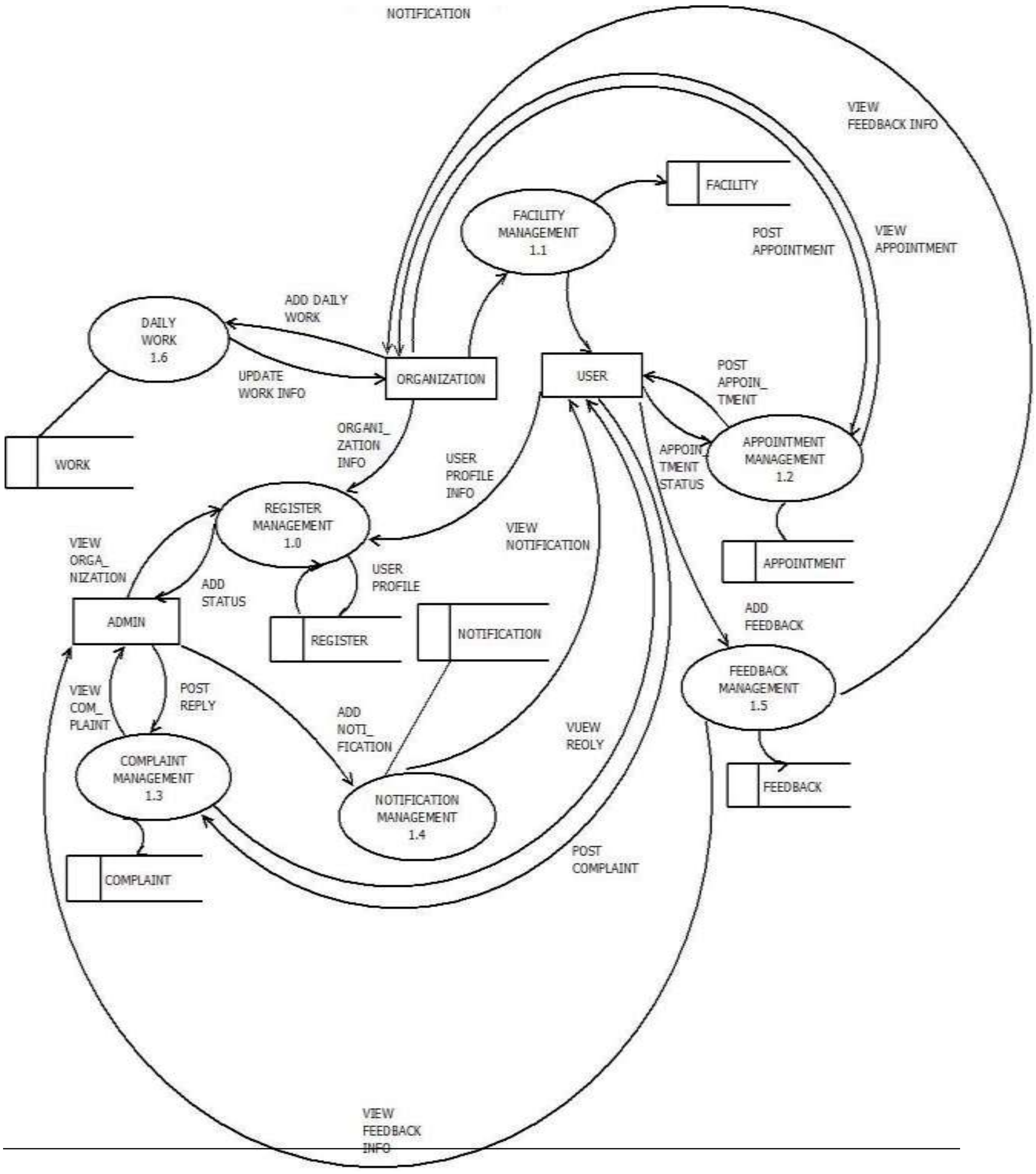
External Entities

External entities are outside the system but they either supply input the system or use other systems output. They are entities on which the designer has control. External entities that supply data into the system are sometimes called source. External entities that use the system data are called sinks. These are represented by rectangles in the data flow diagram.

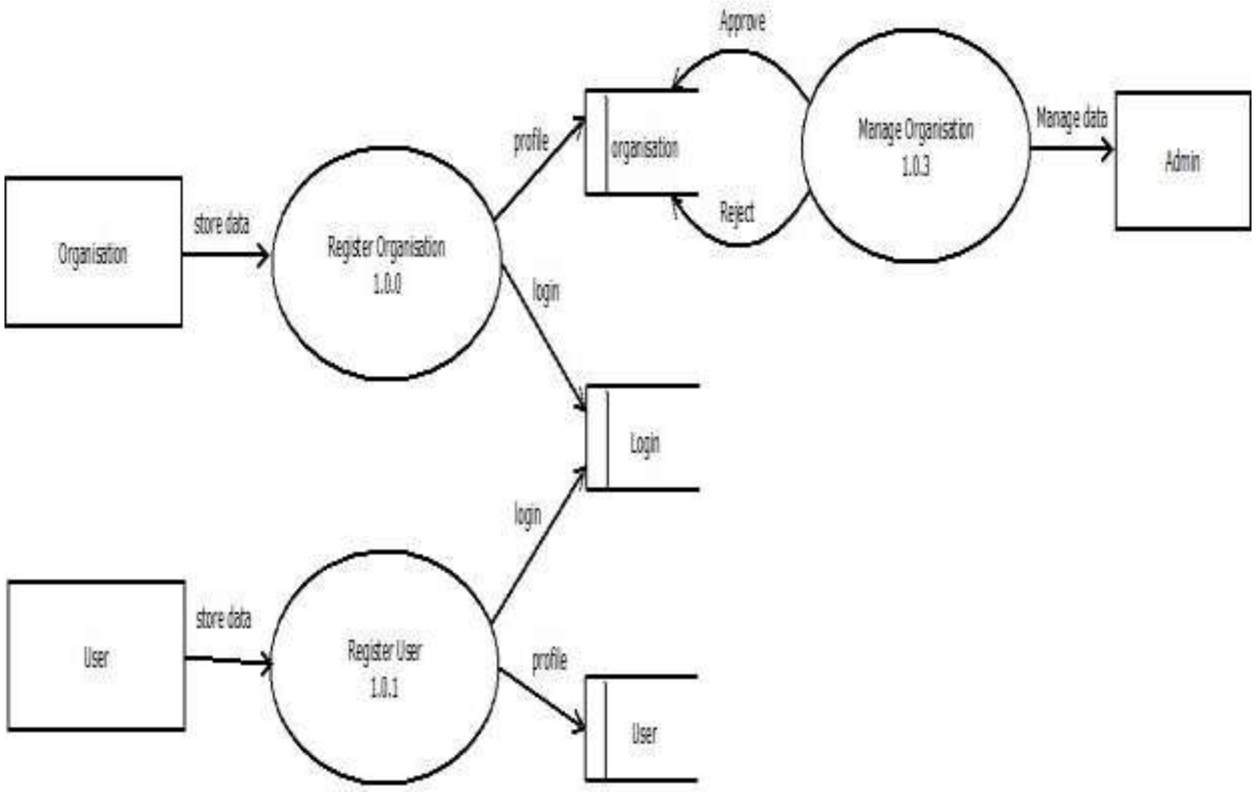
LEVEL 0


Admin
store data
retrieve data
Organisation
store data
retrieve data
Zero Queue
store data
retrieve data
DataBase
store data
retrieve data
User
retrieve data

LEVEL 1

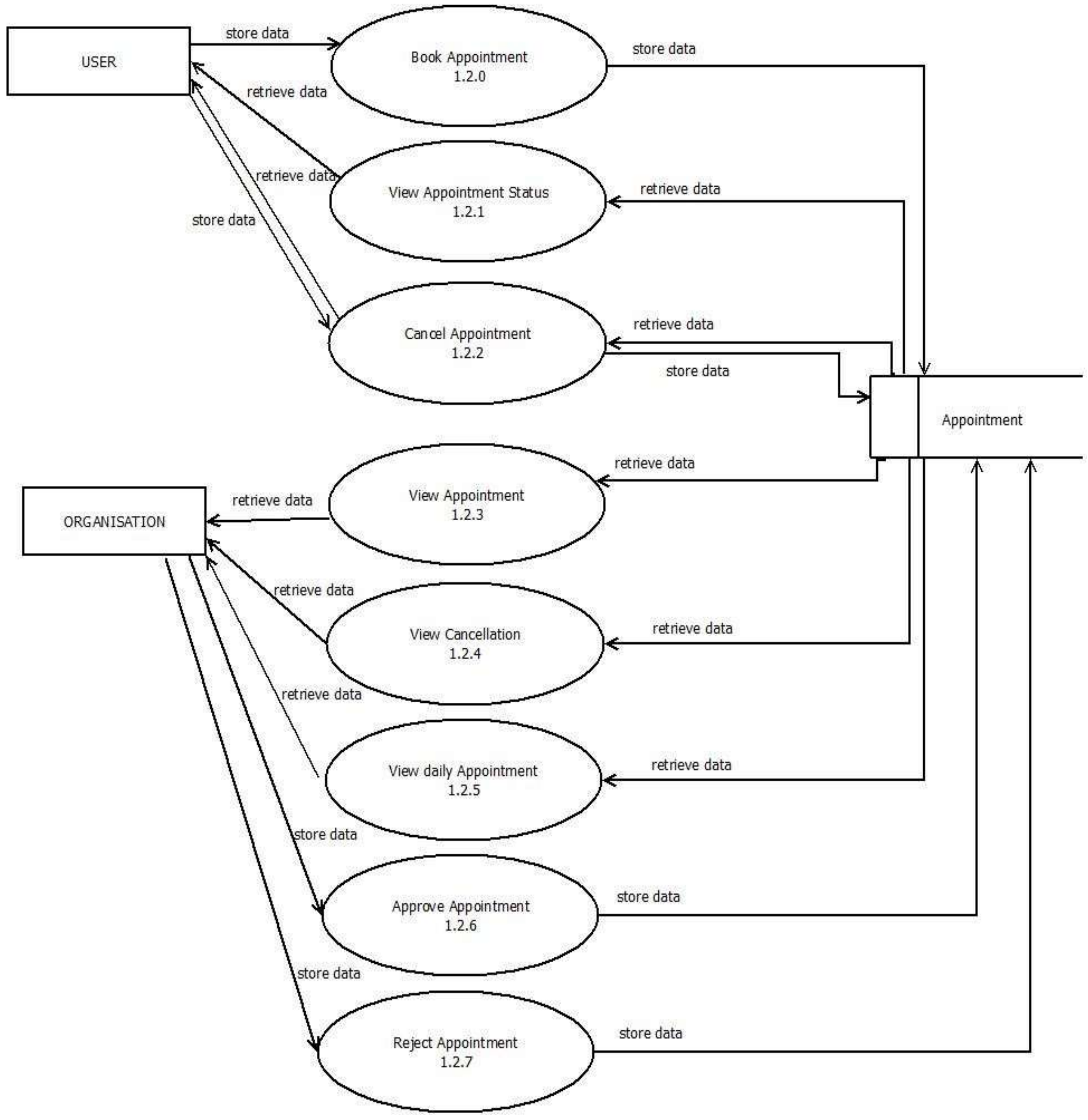

NOTIFICATION
VIEW FEEDBACK INFO
FACILITY
FACILITY MANAGEMENT 1.1
POST APPOINTMENT
VIEW APPOINTMENT
ADD DAILY WORK
DAILY WORK 1.6
UPDATE WORK INFO
ORGANIZATION
USER
POST APPOIN_ TMENT
WORK
ORGANI_ ZATION INFO
USER PROFILE INFO
APPOIN_ TMENT STATUS
APPOINTMENT MANAGEMENT 1.2
REGISTER MANAGEMENT 1.0
VIEW NOTIFICATION
APPOINTMENT
VIEW ORGA_ NIZATION
ADD STATUS
USER PROFILE
ADMIN
REGISTER
NOTIFICATION
ADD FEEDBACK
FEEDBACK MANAGEMENT 1.5
VIEW COM_ PLAINT
POST REPLY
ADD NOTI_ FICATION
VUEW REOLY
COMPLAINT MANAGEMENT 1.3
NOTIFICATION MANAGEMENT 1.4
FEEDBACK
COMPLAINT
POST COMPLAINT
VIEW FEEDBACK INFO

LEVEL 1.0

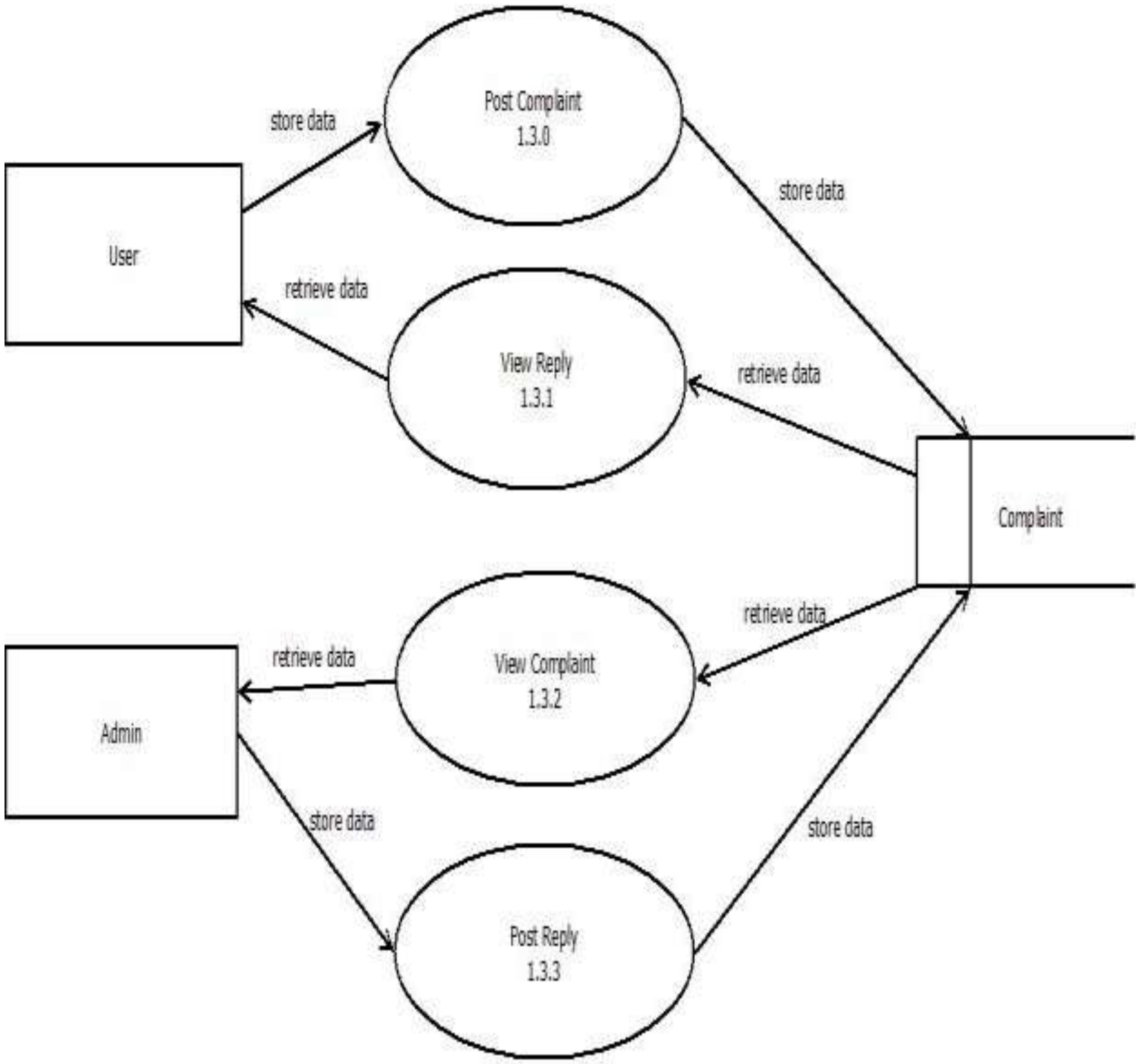


Approve
Manage Organisation 1.0.3
Manage data
Admin
profile
organisation
Reject
Organisation
store data
Register Organisation 1.0.0
login
Login
login
User
store data
Register User 1.0.1
profile
User

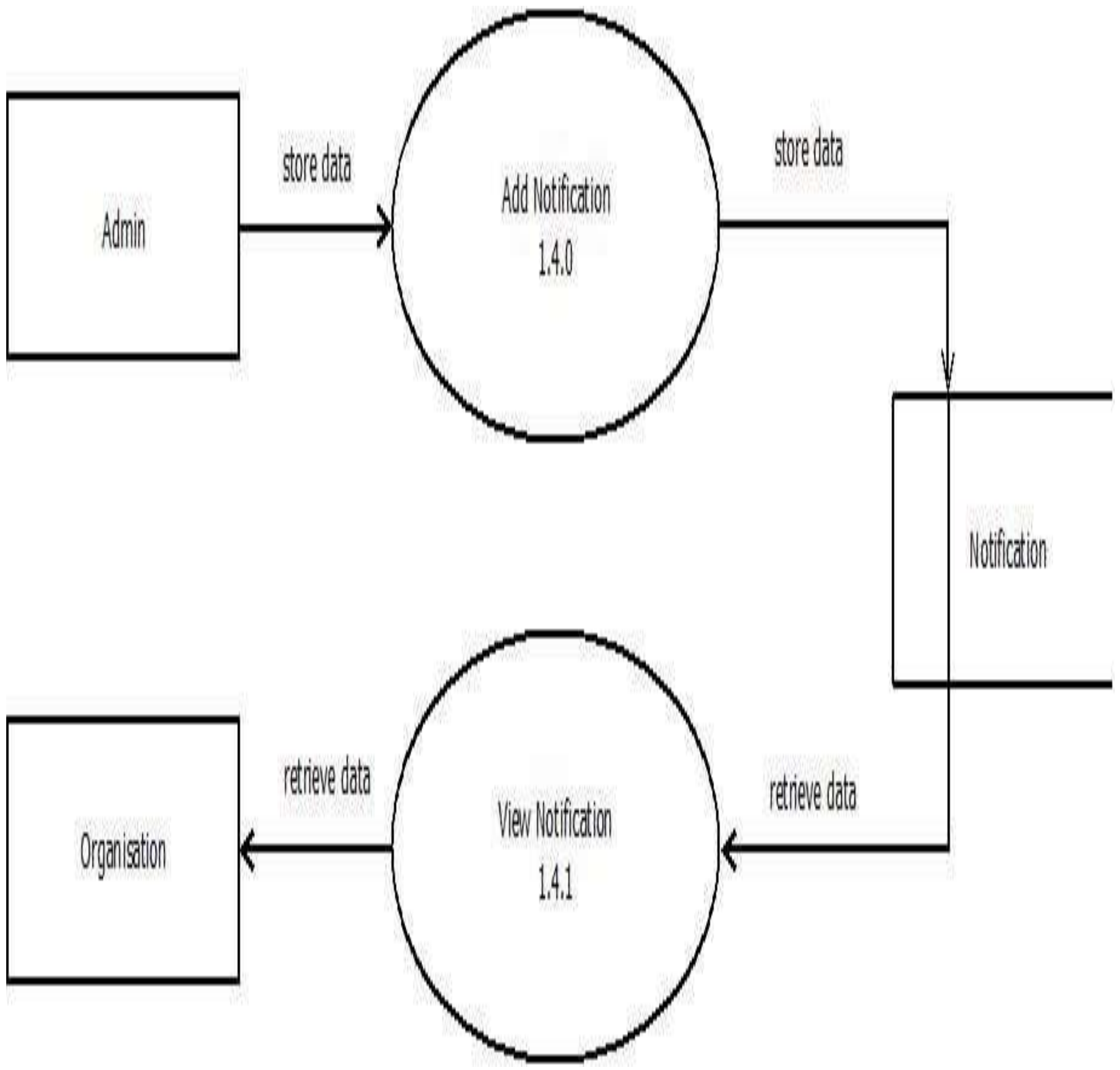
LEVEL 1.2


USER
store data
retrieve data
retrieve data
store data
Book Appointment 1.2.0
View Appointment Status 1.2.1
Cancel Appointment 1.2.2
store data
retrieve data
retrieve data
store data
Appointment
ORGANISATION
retrieve data
View Appointment 1.2.3
retrieve data
retrieve data
View Cancellation 1.2.4
retrieve data
retrieve data
View daily Appointment 1.2.5
retrieve data
store data
Approve Appointment 1.2.6
store data
store data
Reject Appointment 1.2.7
store data

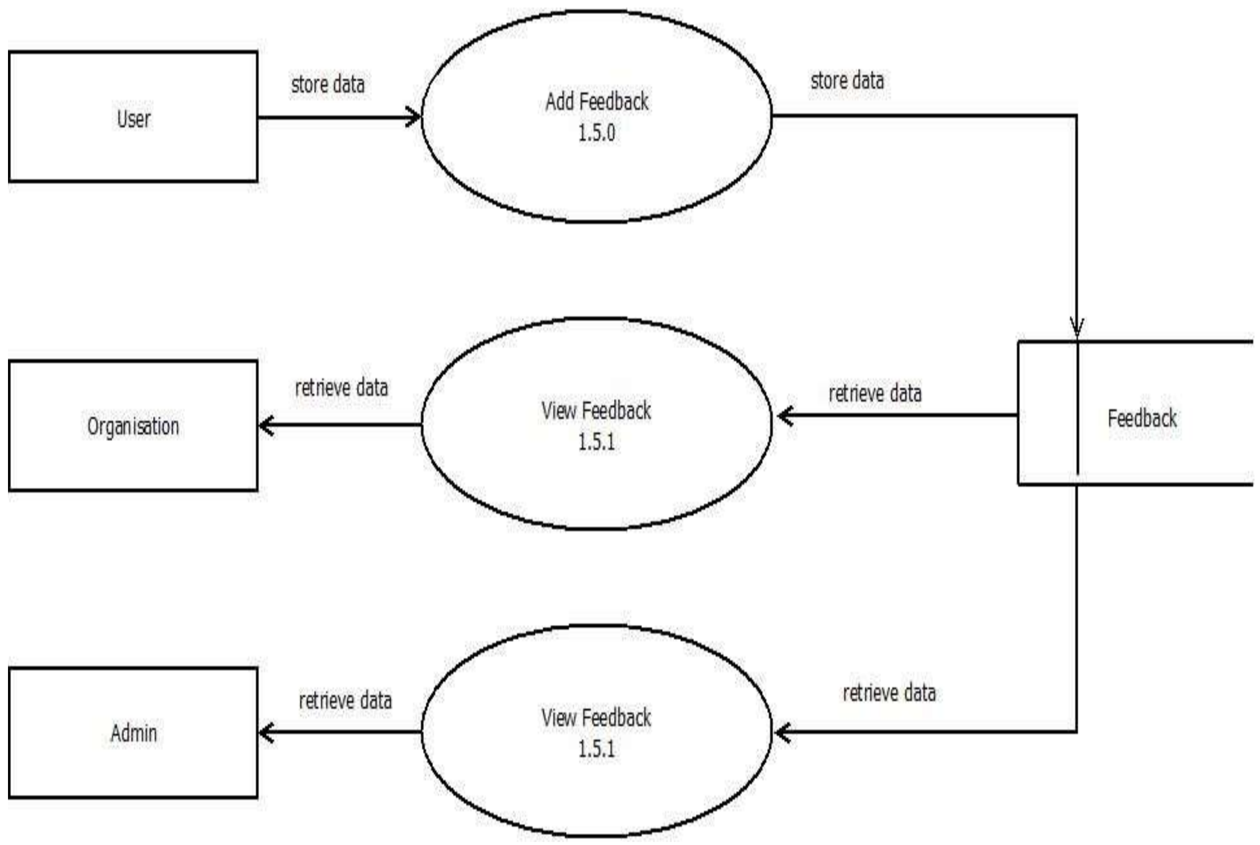
LEVEL 1.3


store data
Post Complaint
1.3.0
store data
User
retrieve data
View Reply
1.3.1
retrieve data
Complaint
retrieve data
retrieve data
View Complaint
1.3.2
Admin
store data
store data
Post Reply
1.3.3

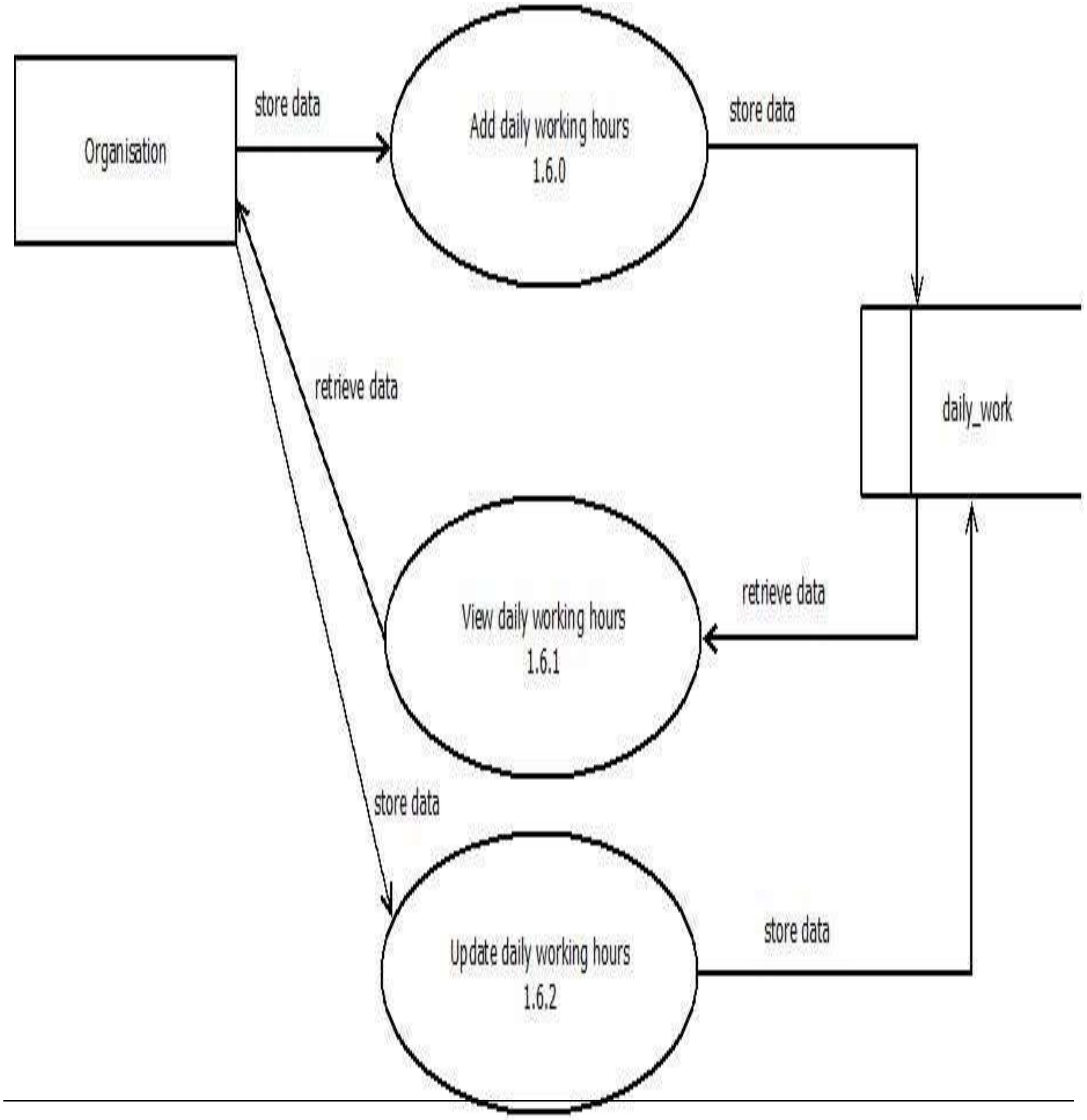
LEVEL 1.4


Admin
store data
Add Notification
1.4.0
store data
Notification
Organisation
retrieve data
View Notification
1.4.1
retrieve data

LEVEL 1.5


User
store data
Add Feedback
1.5.0
store data
Organisation
retrieve data
View Feedback
1.5.1
retrieve data
Feedback
Admin
retrieve data
View Feedback
1.5.1
retrieve data

LEVEL 1.6

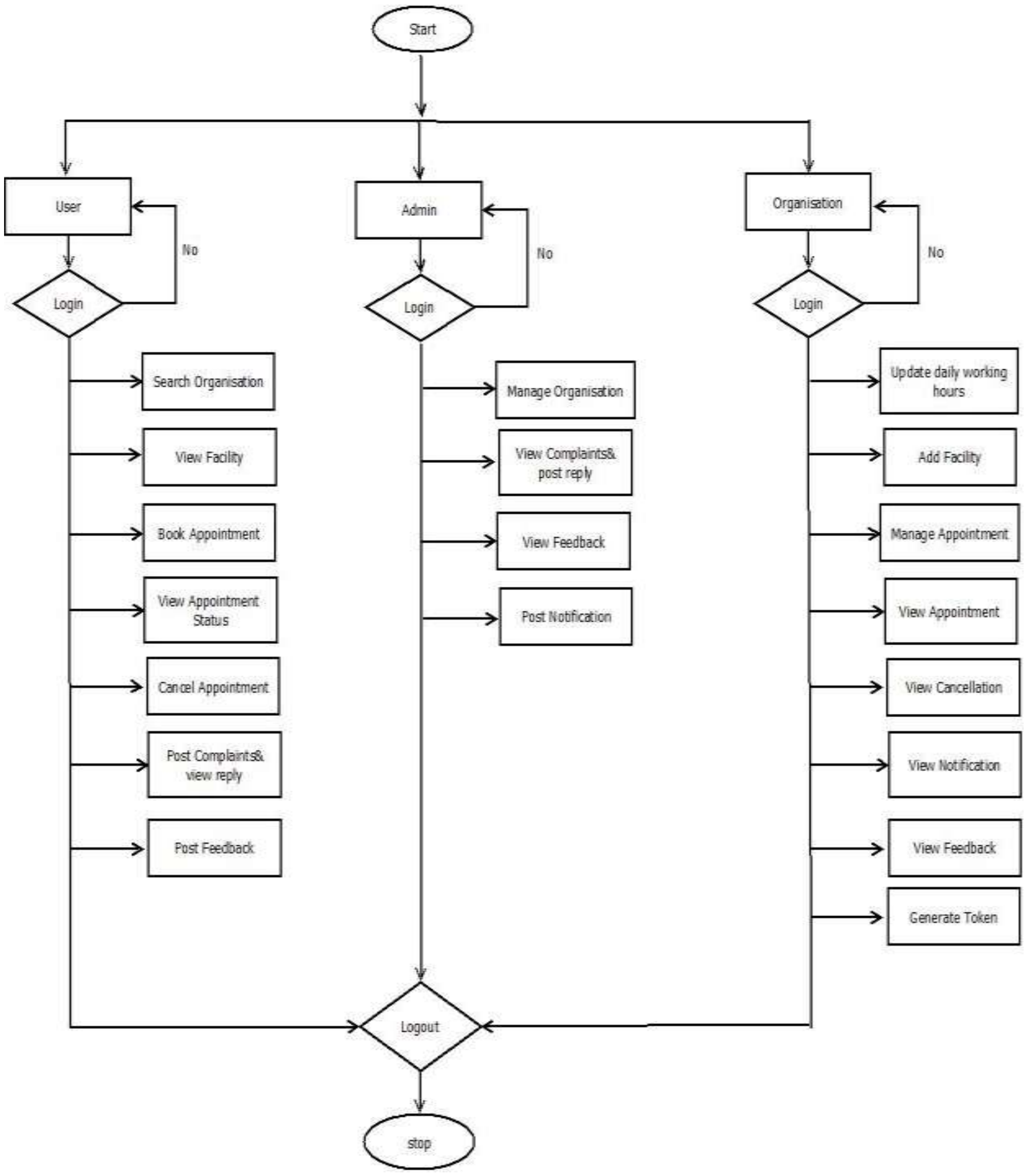

Organisation
store data
Add daily working hours
1.6.0
store data
daily_work
retrieve data
retrieve data
View daily working hours
1.6.1
store data
Update daily working hours
1.6.2
store data

2. CONTROL FLOW DIAGRAM

A control-flow diagram can consist of a subdivision to show sequential steps, with if-then- else conditions, repetition, and/or case conditions. Suitably annotated geometrical figures are used to represent operations, data, or equipment, and arrows are used to indicate the sequential flow from one to another.

- There are several types of control-flow diagrams, for example
- Change-control-flow diagram, used in project management
- Configuration-decision control-flow diagram, used in configuration management
- Process-control-flow diagram, used in process management
- Quality-control-flow diagram, used in quality control.

In software and systems development, control-flow diagrams can be used in control-flow analysis, data-flow analysis, algorithm analysis, and simulation. Control and data are most applicable for real time and data-driven systems. These flow analyses transform logic and data requirements text into graphic flows which are easier to analyse than the text. PERT, state transition, and transaction diagrams are examples of control-flow diagrams.


Start
User
Admin
Organisation
Login
No
Login
No
Login
No
Search Organisation
View Facility
Book Appointment
View Appointment Status
Cancel Appointment
Post Complaints& view reply
Post Feedback
Manage Organisation
View Complaints& post reply
View Feedback
Post Notification
Update daily working hours
Add Facility
Manage Appointment
View Appointment
View Cancellation
View Notification
View Feedback
Generate Token
Logout
stop

3. SEQUENCE DIAGRAM

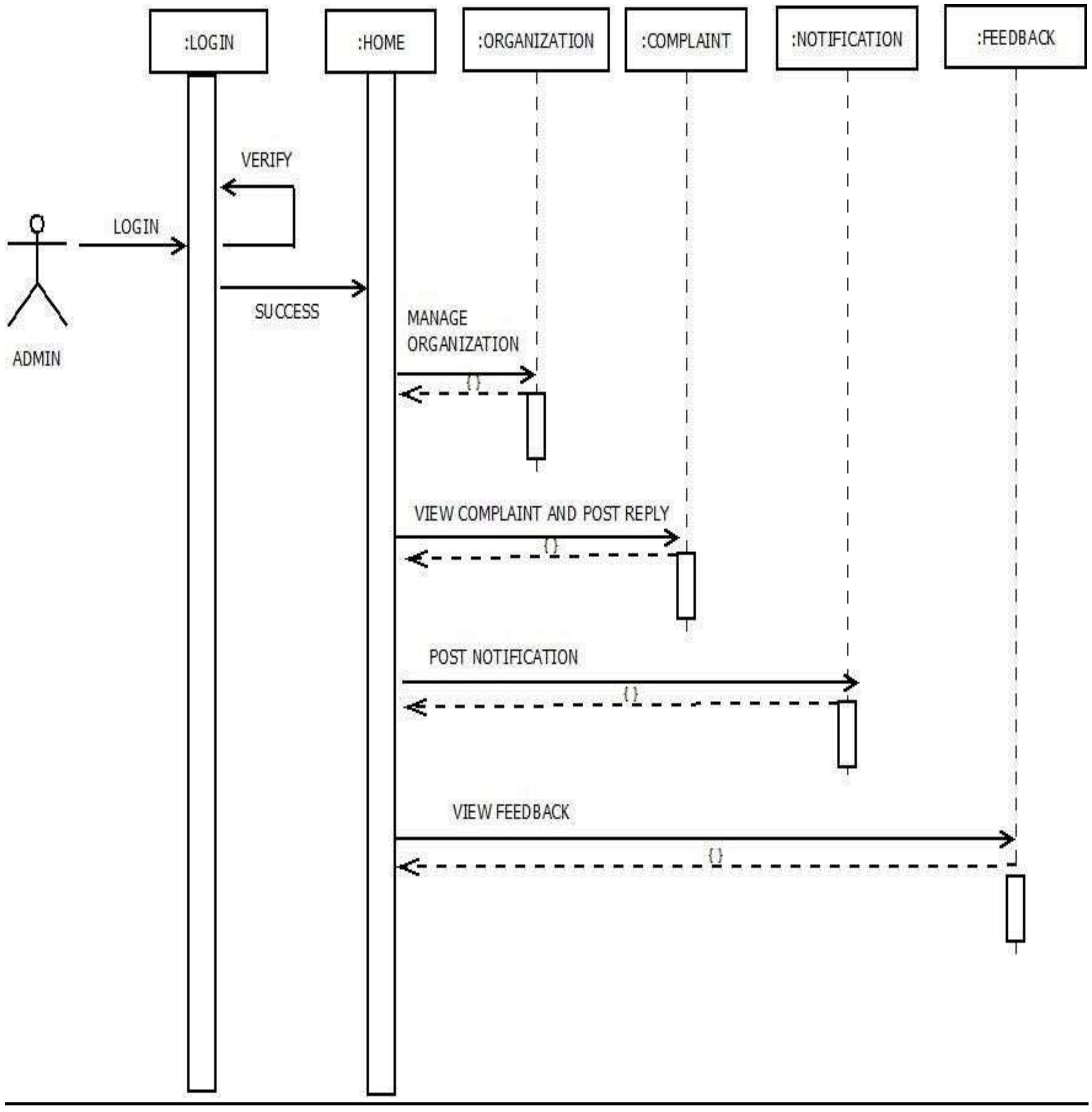
In this post we discuss Sequence Diagrams. Unified Modelling Language (UML) is a modelling language in the field of software engineering which aims to set standard ways to visualize the design of a system. UML guides the creation of multiple types of diagrams such as interaction, structure and behaviour diagrams.

A sequence diagram is the most commonly used interaction diagram.

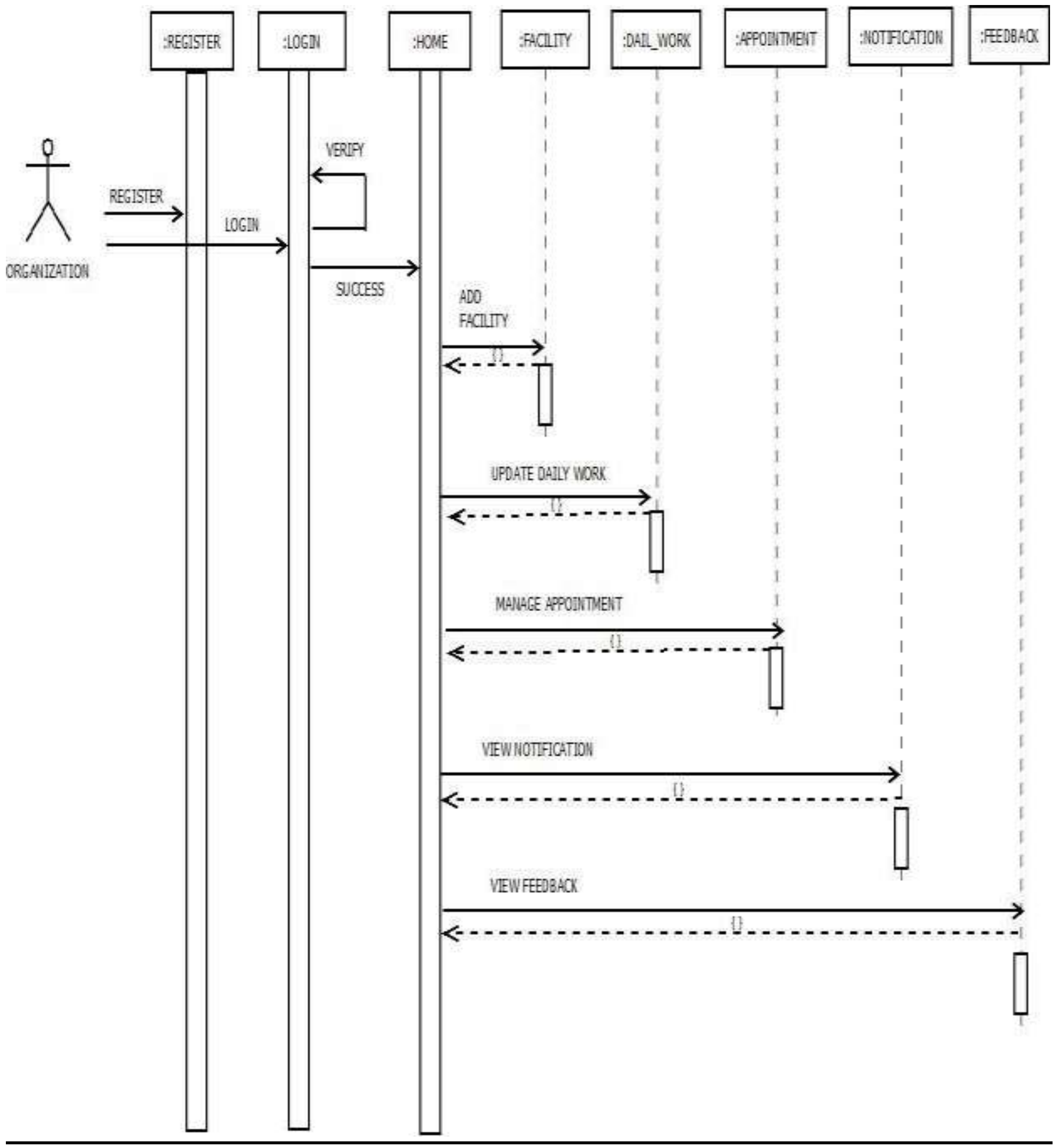
Interaction diagram –An interaction diagram is used to show the interactive behaviour of a system. Since visualizing the interactions in a system can be a cumbersome task, we use different types of interaction diagrams to capture various features and aspects of interaction in a system.

Sequence Diagrams –A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function. These diagrams are widely used by businessmen and software developers to document and understand requirements for new and existing systems.

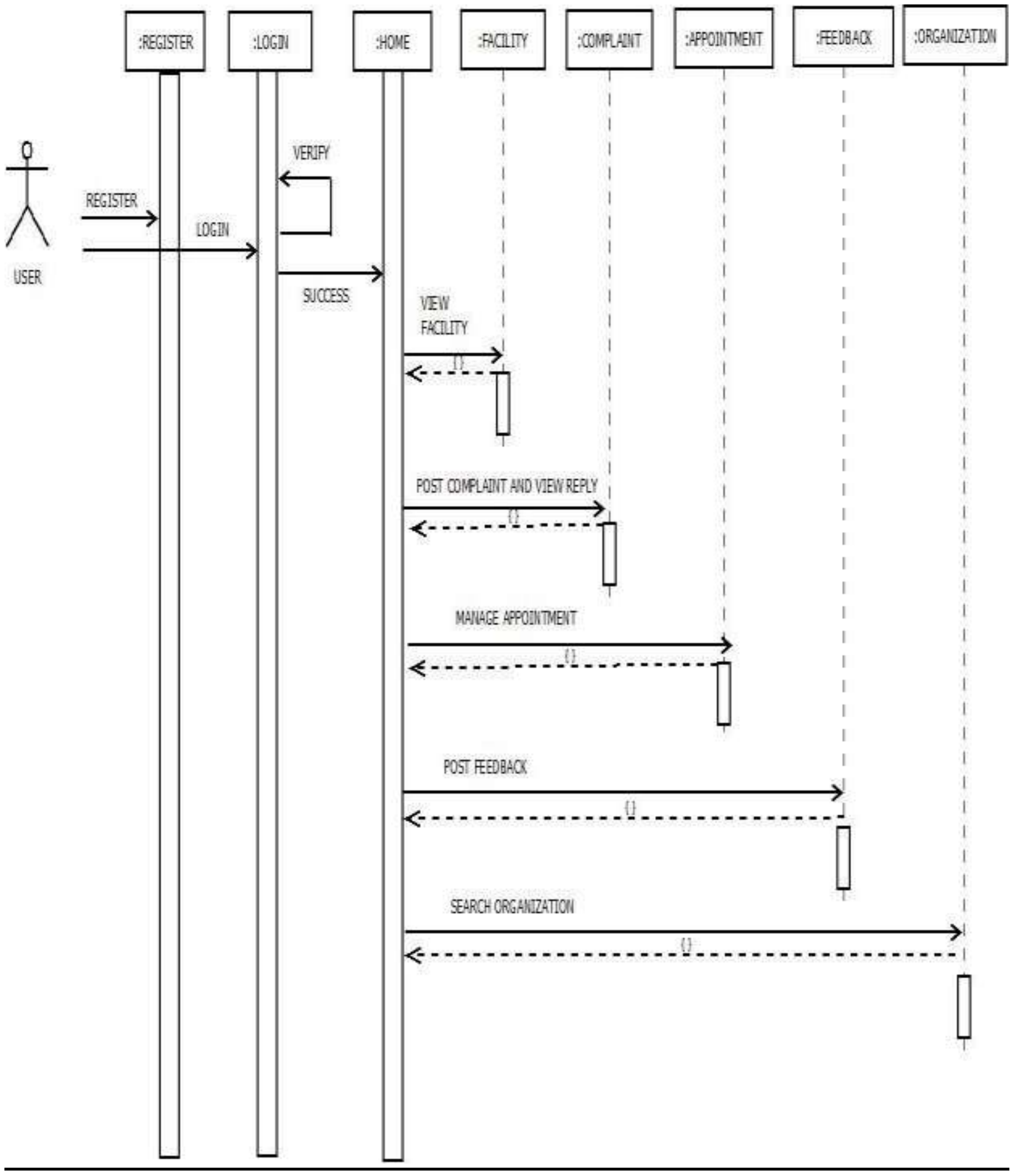
ADMIN


:LOGIN
:HOME
:ORGANIZATION
:COMPLAINT
:NOTIFICATION
:FEEDBACK
VERIFY
LOGIN
ADMIN
SUCCESS
MANAGE ORGANIZATION
{}
VIEW COMPLAINT AND POST REPLY
{}
POST NOTIFICATION
{}
VIEW FEEDBACK
{}

ORGANIZATION



USER



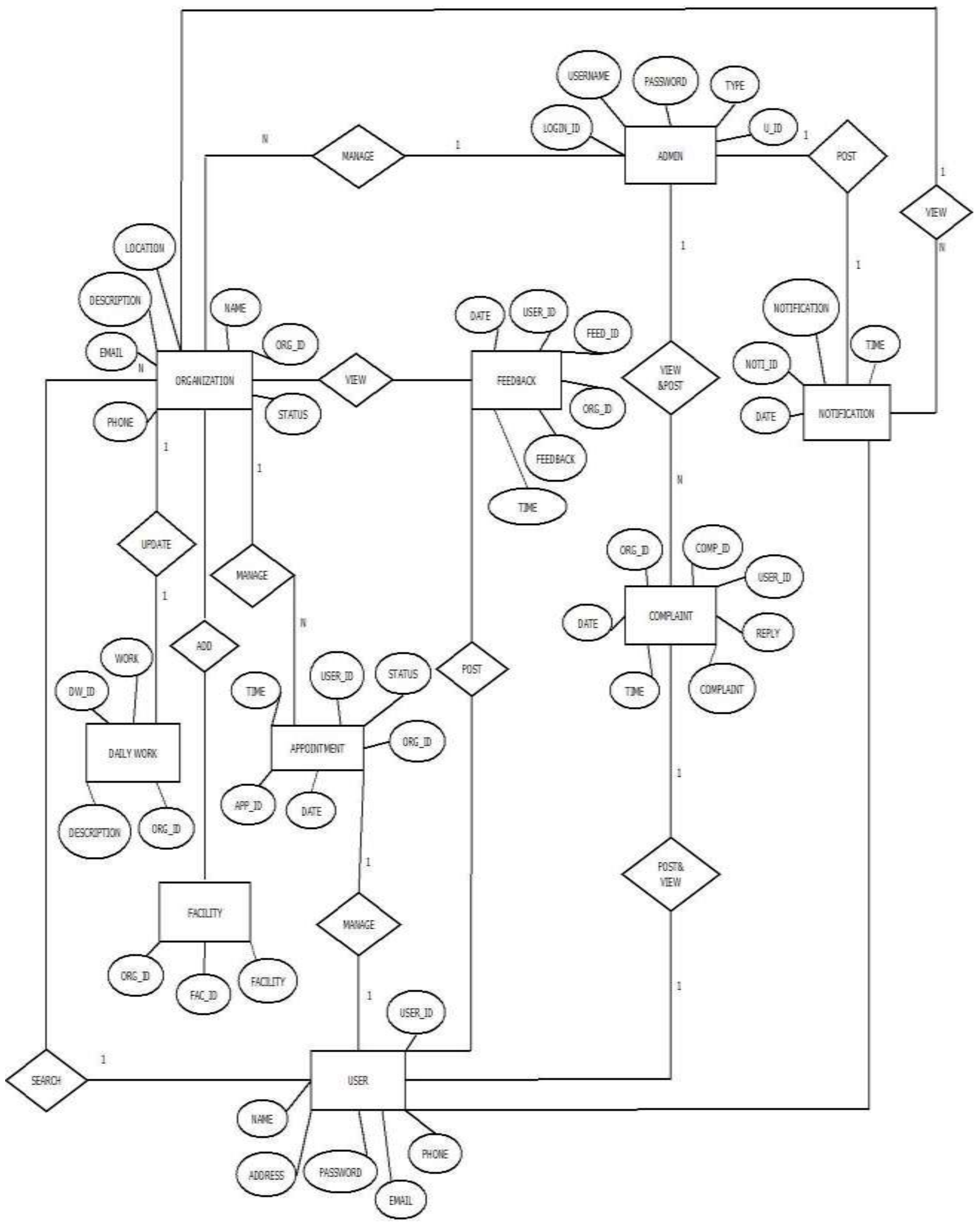
:REGISTER
:LOGIN
:HOME
:FACILITY
:COMPLAINT
:APPOINTMENT
:FEEDBACK
:ORGANIZATION
VERIFY
REGISTER
LOGIN
USER
SUCCESS
VIEW FACILITY
{}
POST COMPLAINT AND VIEW REPLY
{}
MANAGE APPOINTMENT
{}
POST FEEDBACK
{}
SEARCH ORGANIZATION
{}

4. E-R DIAGRAM

An Entity–relationship model (ER model) describes the structure of a database with the help of a diagram, which is known as Entity Relationship Diagram (ER Diagram). An ER model is a design or blueprint of a database that can later be implemented as a database. The main components of E-R model are: entity set and relationship set. An ER diagram shows the relationship among entity sets. An entity set is a group of similar entities and these entities can have attributes. In terms of DBMS, an entity is a table or attribute of a table in database, so by showing relationship among tables and their attributes, ER diagram shows the complete logical structure of a database. Lets have a look at a simple ER diagram to understand this concept.

ER diagram has three main components:

1.Entity
2. Attribute
3. Relationship


USERNAME
PASSWORD
TYPE
LOGIN_ID
U_ID
ADMIN
MANAGE
POST
VIEW
LOCATION
DESCRIPTION
NAME
ORG_ID
EMAIL
ORGANIZATION
VIEW
PHONE
STATUS
DATE
USER_ID
FEED_ID
FEEDBACK
ORG_ID
FEEDBACK
TIME
VIEW &POST
NOTIFICATION
TIME
NOTI_ID
DATE
NOTIFICATION
UPDATE
MANAGE
ADD
POST
ORG_ID
COMP_ID
USER_ID
DATE
COMPLAINT
REPLY
TIME
COMPLAINT
WORK
DW_ID
DAILY WORK
DESCRIPTION
ORG_ID
TIME
USER_ID
STATUS
APPOINTMENT
ORG_ID
APP_ID
DATE
POST& VIEW
FACILITY
MANAGE
ORG_ID
FAC_ID
FACILITY
USER_ID
SEARCH
USER
NAME
ADDRESS
PASSWORD
EMAIL
PHONE
N
1
N

5. USE CASE DIAGRAM

To model a system, the most important aspect is to capture the dynamic behaviour. Dynamic behaviour means the behaviour of the system when it is running/operating. Only static behaviour is not sufficient to model a system rather dynamic behaviour is more important than static behaviour. In UML, there are five diagrams available to model the dynamic nature and use case diagram is one of them. Now as we have to discuss that the use case diagram is dynamic in nature, there should be some internal or external factors for making the interaction. These internal and external agents are known as actors. Use case diagrams consists of actors, use cases and their relationships. The diagram is used to model the system/subsystem of an application. A single use case diagram captures a particular functionality of a system. Hence to model the entire system, a number of use case diagrams are used.

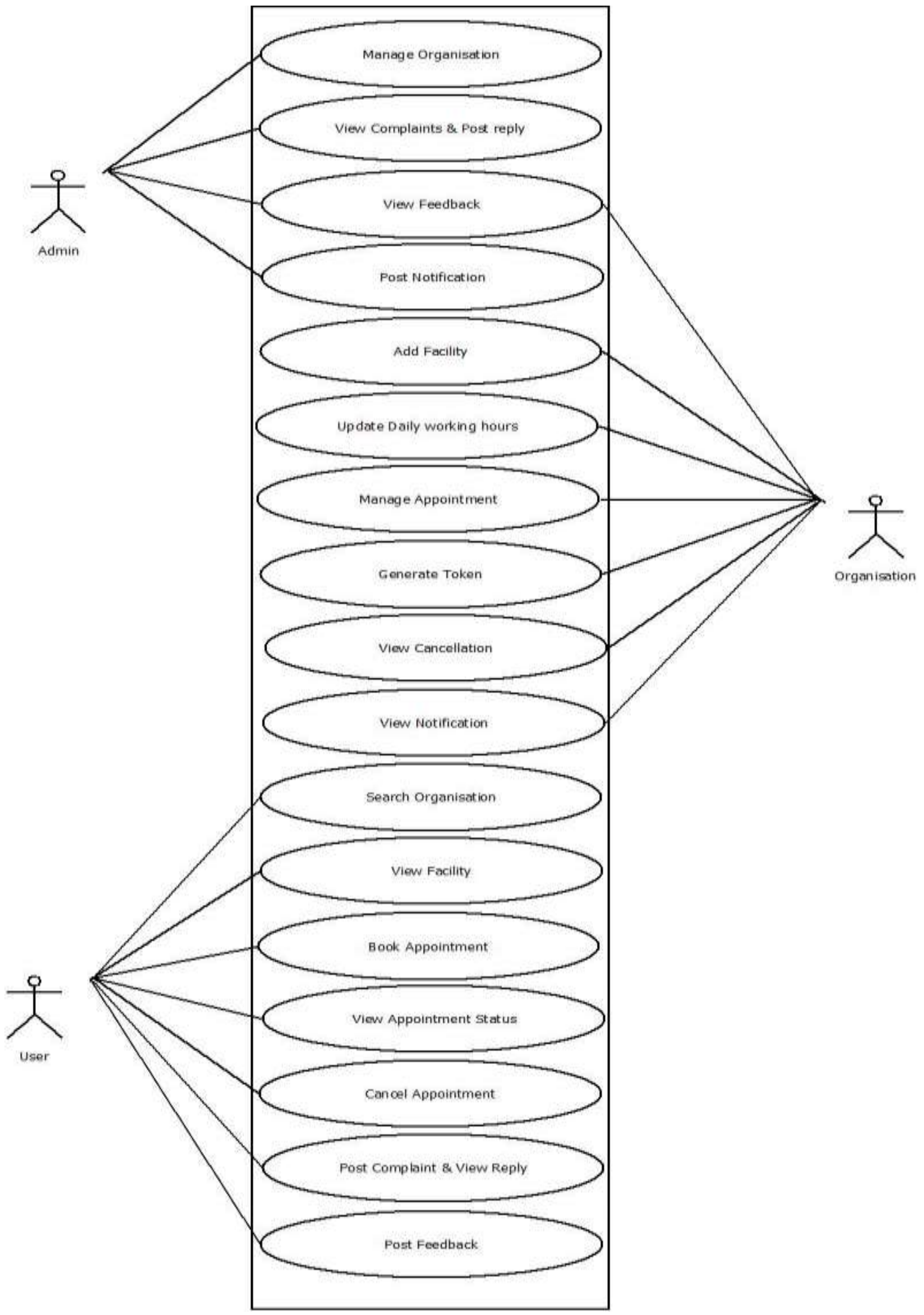
In brief, the purposes of use case diagrams can be said to be as follows –

- Used to gather the requirements of a system.

- Used to get an outside view of a system.

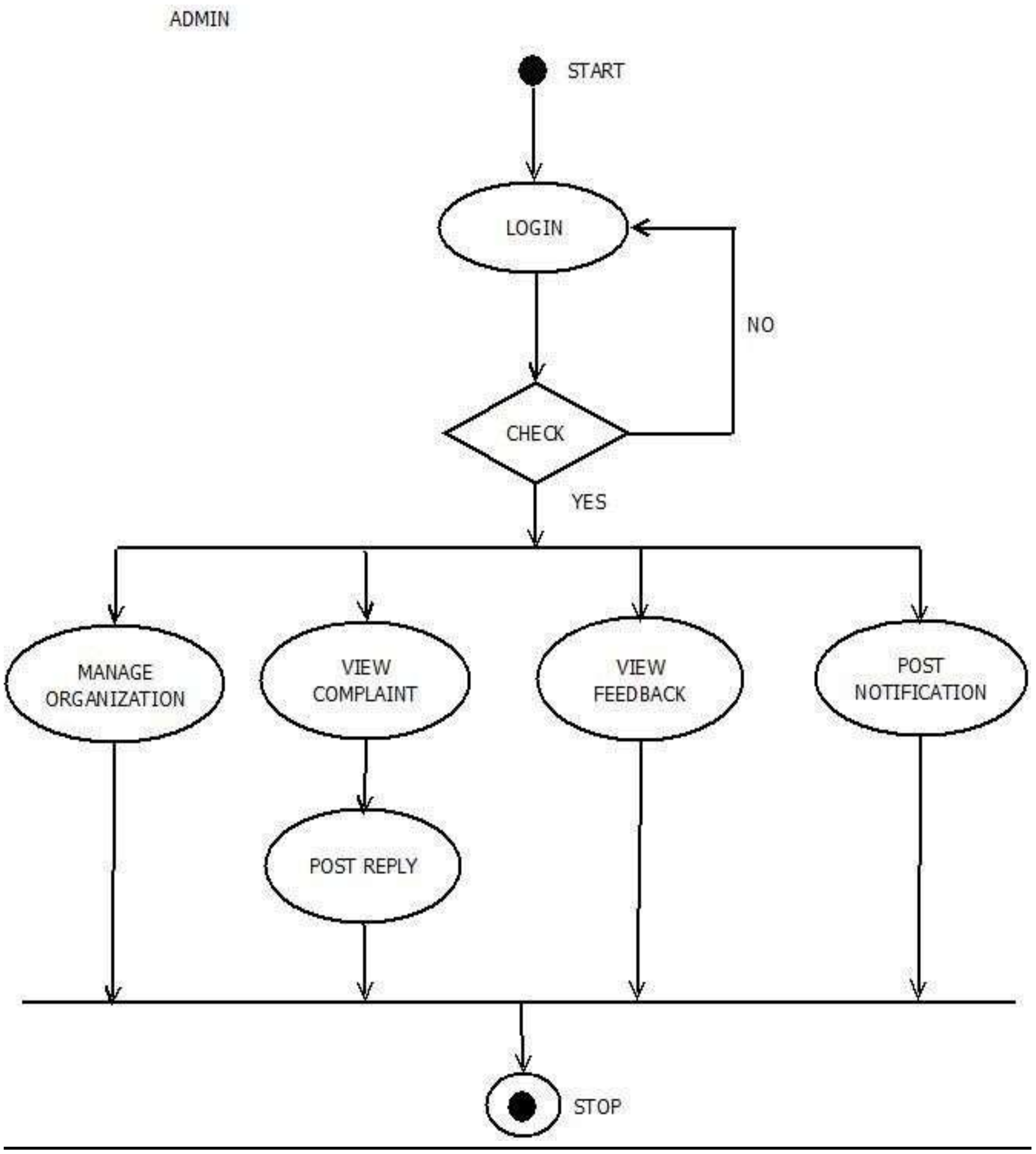
- Identify the external and internal factors influencing the system.

- Show the interaction among the requirements are actors.

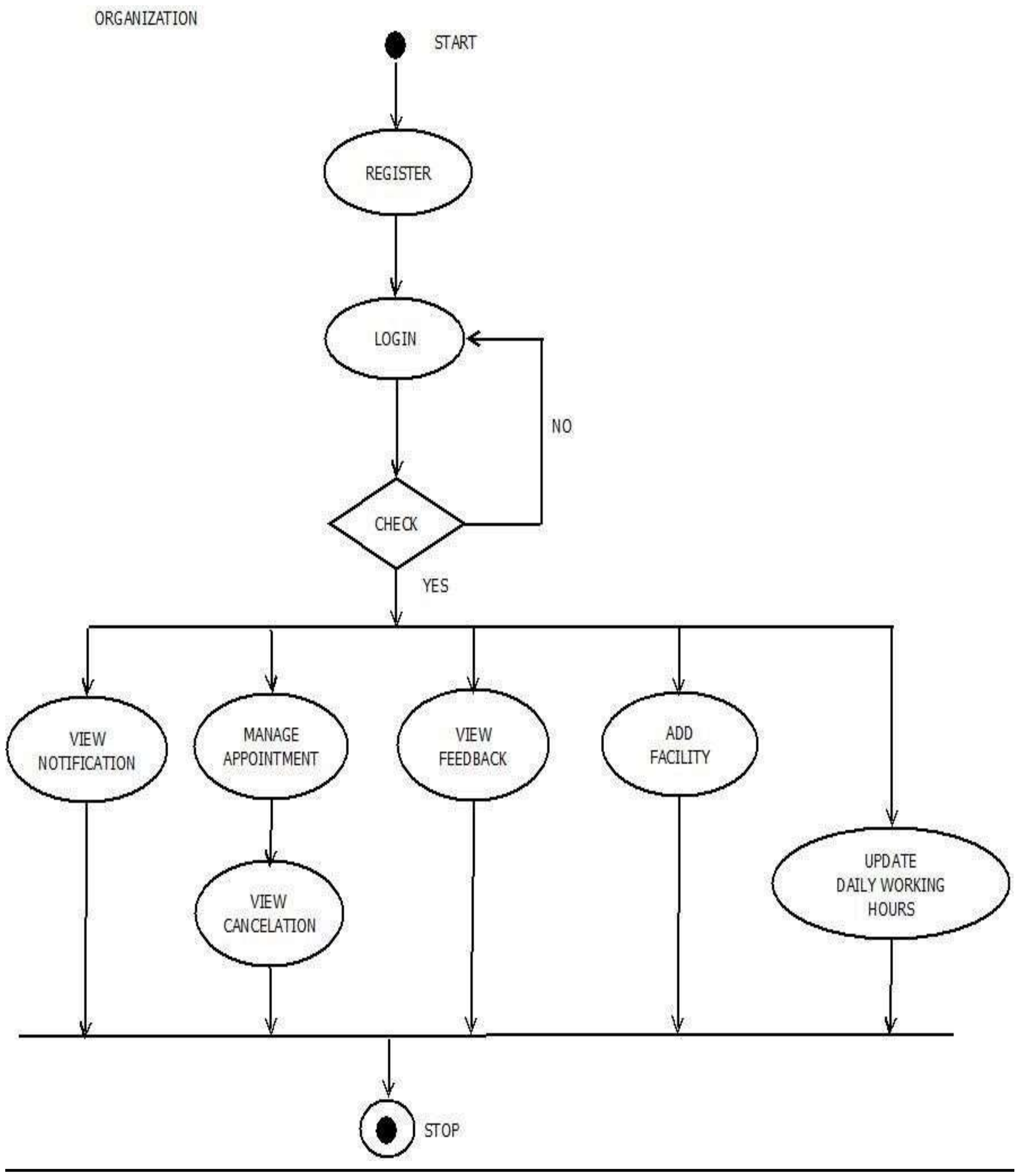

Manage Organisation
View Complaints & Post reply
View Feedback
Post Notification
Add Facility
Update Daily working hours
Manage Appointment
Generate Token
View Cancellation
View Notification
Search Organisation
View Facility
Book Appointment
View Appointment Status
Cancel Appointment
Post Complaint & View Reply
Post Feedback
Admin
Organisation
User

6.ACTIVITY DIAGRAM

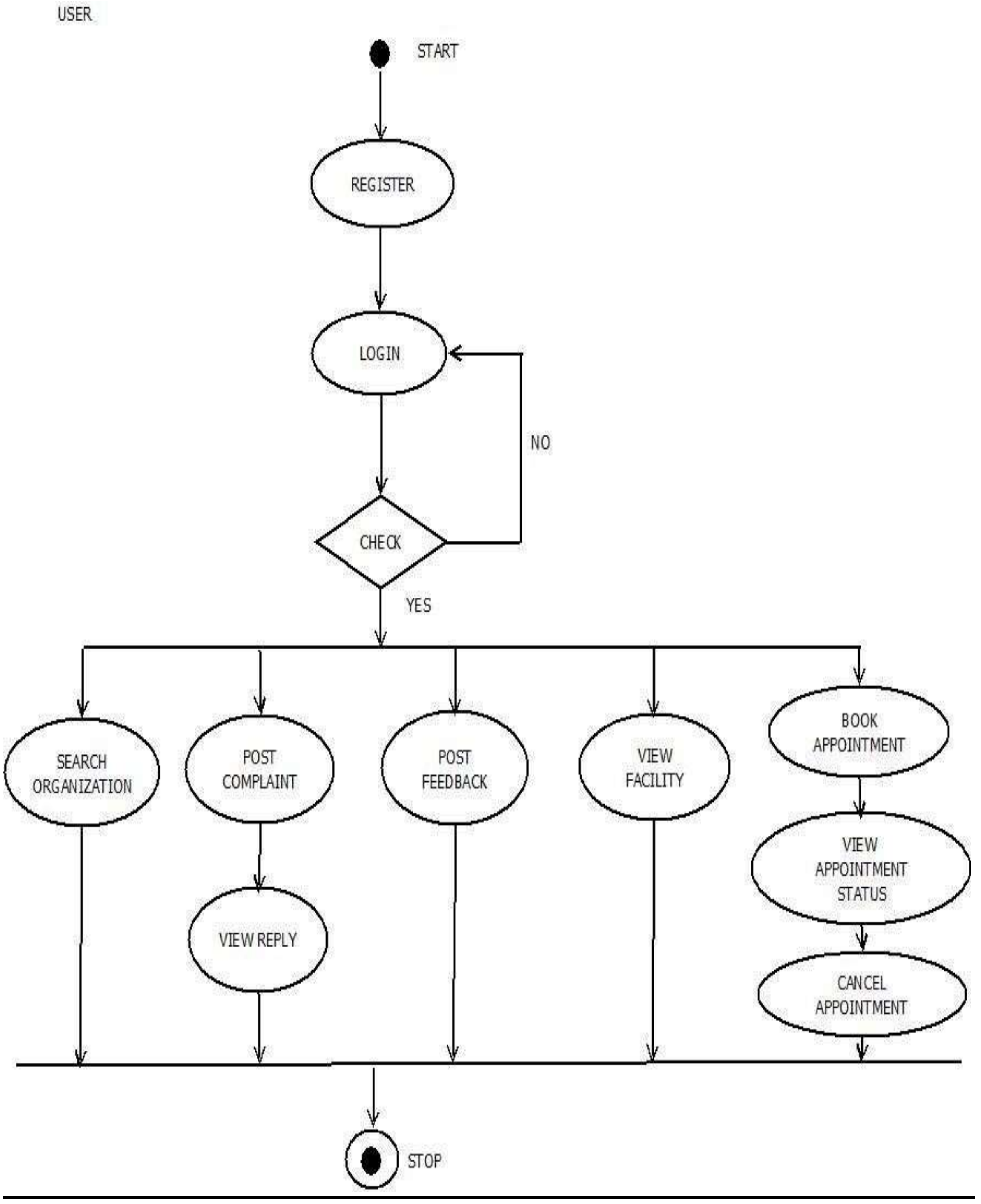
ADMIN



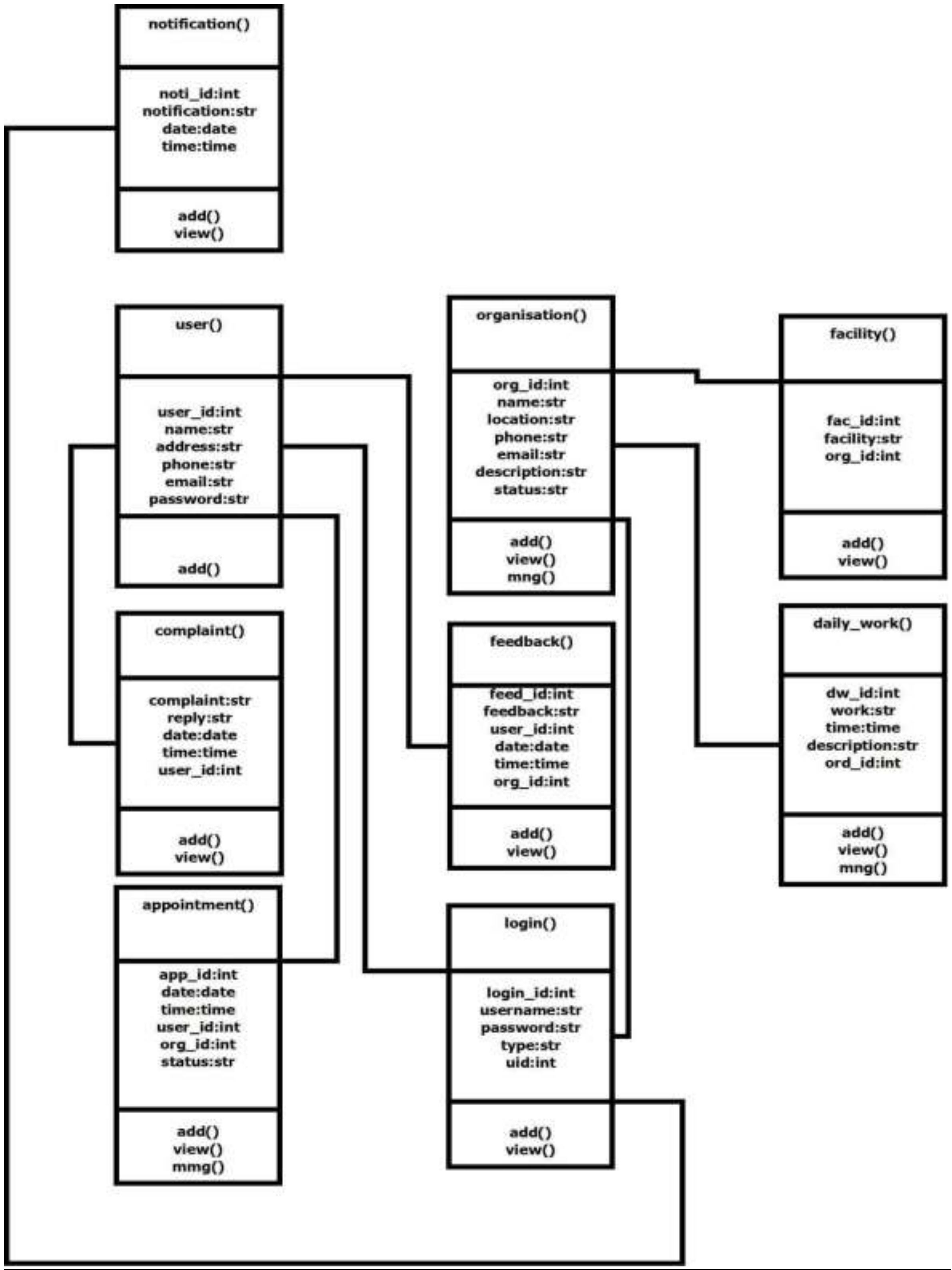
ORGANIZATION



USER


USER
START
REGISTER
LOGIN
NO
CHECK
YES
SEARCH ORGANIZATION
POST COMPLAINT
POST FEEDBACK
VIEW FACILITY
BOOK APPOINTMENT
VIEW REPLY
VIEW APPOINTMENT STATUS
CANCEL APPOINTMENT
STOP

CLASS DIAGRAM



SYSTEM DESIGN

Systems design is the process of defining elements of a system like modules, architecture, components and their interfaces and data for a system based on the specified requirements. It is the process of defining, developing and designing systems which satisfies the specific needs and requirements of a business or organization.
A systemic approach is required for a coherent and well-running system. Bottom-Up or Top-Down approach is required to take into account all related variables of the system. A designer uses the modelling languages to express the information and knowledge in a structure of system that is defined by a consistent set of rules and definitions. The designs can be defined in graphical or textual modelling languages.

Some of the examples of graphical modelling languages are

- Unified Modelling Language (UML): To describe software both structurally and behaviorally with graphical notation.
- Flowchart: A schematic or stepwise representation of an algorithm.
- Business Process Modelling Notation (BPMN): Used for Process Modelling language.
- Systems Modelling Language (SysML): Used for systems engineering.

Design methods:

- Architectural design: To describes the views, models, behavior, and structure of the system.
- Logical design: To represent the data flow, inputs and outputs of the system. Example: ER Diagrams (Entity Relationship Diagrams).

- Physical design: Defined as a) How users add information to the system and how the system represents information back to the user. b) How the data is modelled and stored within the system. c) How data moves through the system, how data is validated, secured and/or transformed as it flows through and out of the system.

MODULARISSATION DETAILS

Modularization is a technique to divide a software system into multiple discrete and independent modules, which are expected to be capable of carrying out task(s) independently. These modules may work as basic constructs for the entire software. Designers tend to design modules such that they can be executed and/or compiled separately and independently.
Modular design unintentionally follows the rules of ‘divide and conquer’ problem-solving strategy this is because there are many other benefits attached with the modular design of a software.

Advantage of modularization:

- Smaller components are easier to maintain
- Program can be divided based on functional aspects
- Desired level of abstraction can be brought in the program
- Components with high cohesion can be re-used again
- Concurrent execution can be made possible
- Desired from security aspect

This topic of computer science permits manipulation and classification of huge amounts of data. C4.5 and ID3 decision tree, for example, have been proven to be efficient for specific prediction cases. This article shows the construction of a predictive model of student dropout, in order to predict the probability that a student drop out his/her an academic program, by means of two data mining techniques and comparison of results. There has been a growing interest and concern about the problem of school failure and determining the main factors contributing to this problem in recent years. The number of students who are dropping out each year has been increasing. Therefore, many researches were aimed at examining the factors that affect the low performance of students at different educational levels like primary, secondary and higher. The amount of information stored in educational databases is rapidly increasing due to the advancement in the field of information technology. These databases contain a wealth of data about students and are a gold mine of valuable information. The difficult task here is to identify and classify the valuable information hidden in those databases. A very promising solution for this problem is the use of C4.5 and ID3 decision tree with the use of for knowledge discovery in databases or data mining from a given set of attributes

MODULES

TECHNICAL

- Booking management
- Token generator
- Waiting list
- Cancellation
- Notification

ADMIN

- Login
- Manage organization
- View complaints and post reply
- View feedback
- Post notification

Organization

- Register
- Login
- Add facilities
- Update daily working hours
- View appointments and manage
- Generate token
- View cancellation,View Notification
- View daily appointments
- View feedback

USER

- Register
- Login
- Search organization
- View facilities
- Book appointment
- View appointment status
- Cancel appointment
- Post complaints and view reply
- Post feedback

DATA INTEGRITY AND CONSTRAINTS

Data integrity is a fundamental component of information security. In its broadest use, “data integrity” refers to the accuracy and consistency of data stored in a database, data warehouse, data mart or other construct. The term – Data Integrity - can be used to describe a state, a process or a function – and is often used as a proxy for “data quality”. Data with “integrity” is said to have a complete or whole structure. Data values are standardized according to a data model and/or data type. All characteristics of the data must be correct – including business rules, relations, dates, definitions and lineage – for data to be complete. Data integrity is imposed within a database when it is designed and is authenticated through the ongoing use of error checking and validation routines. As a simple example, to maintain data integrity numeric columns/cells should not accept alphabetic data.

Logical Integrity

Logical integrity keeps data unchanged as it’s used in different ways in a relational database. Logical integrity protects data from human error and hackers as well, but in a much different way than physical integrity does. There are four types of logical integrity.

Entity Integrity

Entity integrity relies on the creation of primary keys, or unique values that identify pieces of data, to ensure that data isn’t listed more than once and that no field in a table is null. It’s a feature of relational systems which store data in tables that can be linked and used in a variety of ways.

Referential Integrity

Referential integrity refers to the series of processes that make sure data is stored and used uniformly. Rules embedded into the database’s structure about how foreign keys are used ensure that only appropriate changes, additions, or deletions of data occur. Rules may include constraints that eliminate the entry of duplicate data, guarantee that data is accurate, and/or disallow the entry of data that doesn’t apply.

Domain Integrity

Domain integrity is the collection of processes that ensure the accuracy of each piece of data in a domain. In this context, a domain is a set of acceptable values that a column is allowed to contain. It can include constraints and other measures that limit the format, type, and amount of data entered.

User-Defined Integrity

User-defined integrity involves the rules and constraints created by the user to fit their particular needs. Sometimes entity, referential, and domain integrity aren't enough to safeguard data. Often, specific business rules must be taken into account and incorporated into data integrity measures.

DATABASE AND TABLES

Database Design is a collection of processes that facilitate the designing, development, implementation and maintenance of enterprise data management systems. Properly designed database are easy to maintain, improves data consistency and are cost effective in terms of disk storage space. The database designer decides how the data elements correlate and what data must be stored. The main objectives of database designing are to produce logical and physical designs models of the proposed database system. The logical model concentrates on the data requirements and the data to be stored independent of physical considerations. It does not concern itself with how the data will be stored or where it will be stored physically. The physical data design model involves translating the logical design of the database onto physical media using hardware resources and software systems such as database management systems (DBMS).

The objective of database design is to provide auxiliary storage and to contribute to the overall efficiency of the computer program component. One auxiliary storage medium must provide efficient access to the data. The general theme behind a database is to handle information as an integrated whole. The general objective is to make information access easy, quick, inexpensive and flexible for the user. In a database environment, common data are available in which several users can use. The concept behind a database is an integrated collection of data and provides centralized access to the data from the program. It makes possible

to treat data as separate resource. While designing database, several objectives must be considered.
A database is a collection of logically related data stored with minimum redundancy to serve many users quickly and efficiently. Relationships are established between the data items and unnecessary data items are removed. Normalization is done to get an internal consistency of data items and to have minimum redundancy and maximum stability. This ensures minimizing data storage required, minimizing chances of data inconsistencies and optimizing for updates. The Microsoft SQL Server database has been chosen for developing the relevant databases.

NORMALIZATION

The term normalization refers to the way data item are grouped into record structures. In other words, normalization is a technique of separating redundant fields and breaking up large tables into smaller one. After the conceptual level, the next level of process of database design to organize the database structure into a good shape is called normalization.
Normalization is adopted to overcome drawbacks like :

REPARTITION OF DATA LOSS OF INFORMATION INCONSISTENCY

In our design all tables have been normalized up to the third normal form. The different normal forms applied during the database design are given below.

- First Normal Form (1NF)
- Second Normal Form (2NF)
- Third Normal Form (3NF)

First Normal Form (1NF)

A relation is said to be in 1NF if it satisfies the constraints that it contains primary key only. Our application satisfies INF.

Second Normal Form (2NF)

A relation is said to be in 2NF if and only if it satisfies all 1NF conditions for the primary key and every non-primary key attributes of the relation is fully dependent on its primary key alone. If a non-key attribute is not dependent on the key, it should be removed from the relation and places as a separate relation. i.e., the fields of the table in 2NF are all related to primary key.

Third Normal Form (3NF)

A relation is said to be in 3NF if and only if only it is in 2NF and more over non-key attribute of the relation should not depend on other non-key attributes.

The data in the system has to be stored and retrieved from database. Designing the database is a part of system design. Data elements and data structures to be stored have been identified at analysis stage. They are structured and put together to design the data storage and retrieval system.

LOGIN

COLUMN NAME	DATATYPE	LENGHT	CONSTRAINT	DESCRIPTION
Login_id	Int	10	Primary Key	Primary key of the table
Username	Varchar	20	Not Null	Username for login
Password	Varchar	20	Not Null	Password for login
Type	Varchar	20	Not NullType	Type for login
U_id	Int	10	Not Null	U_id for login

ORGANIZATION

COLUMN NAME	DATATYPE	LENGHT	CONSTRAINT	DESCRIPTION
organization_id	Int	10	Primary Key	Primary key of the table
name	Varchar	20	Not Null	Organization name
location	Varchar	20	Not Null	Location of organization
phone	Varchar	10	Not Null	Organization phone
email	Varchar	20	Not Null	Organization email
description	Varchar	30	Not Null	Description about organization
status	Varchar	10	Not Null	Status for manage organization

COMPLAINT

COLUMN NAME	DATATYPE	LENGHT	CONSTRAINT	DESCRIPTION
Complaint_id	Int	10	Primary Key	Primary key of the table
complaint	Varchar	50	Not Null	For post complaints
reply	Varchar	50	Not Null	For post replys
User_id	Int	10	Foreign Key	User id for identify which user’s complaint
date	date	10	Not Null	Complaint date
time	time	10	Not Null	Complaint Time

FEEDBACK

COLUMN NAME	DATATYPE	LENGTH	CONSTRAINT	DESCRIPTION
Feedback_id	Int	10	Primary Key	Primary key of the table
feedback	feedback	50	Not Null	For post feedback
User_id	int	10	Foreign Key	User id for identify which user's feedback
date	date	10	Not Null	Feedback Date
time	time	10	Not Null	Feedback Time

USER

COLUMN NAME	DATATYPE	LENGHT	CONSTRAINT	DESCRIPTION
user_id	Int	10	Primary Key	Primary key of the table
name	Varchar	20	Not Null	User's name
address	Varchar	30	Not Null	User's address
phoneno	Varchar	10	Not Null	User's phone
email	Varchar	20	Not Null	User's email
password	Varchar	20	Not Null	User's password

FACILITY

COLUMN NAME	DATATYPE	LENGHT	CONSTRAINT	DESCRIPTION
Facility_id	Int	10	Primary Key	Primary key of the table
organization_id	Int	10	Not Null	Organization's primary key
facility	Varchar	20	Not Null	For post facility

NOTIFICATION

COLUMN NAME	DATATYPE	LENGHT	CONSTRAINT	DESCRIPTION
notification_id	Int	10	Primary Key	Primary key of the table
Notification	Varchar	50	Not Null	For post notification
date	date	20	Not Null	Notification date
time	time	20	Not Null	Notification time

APPOINMENT

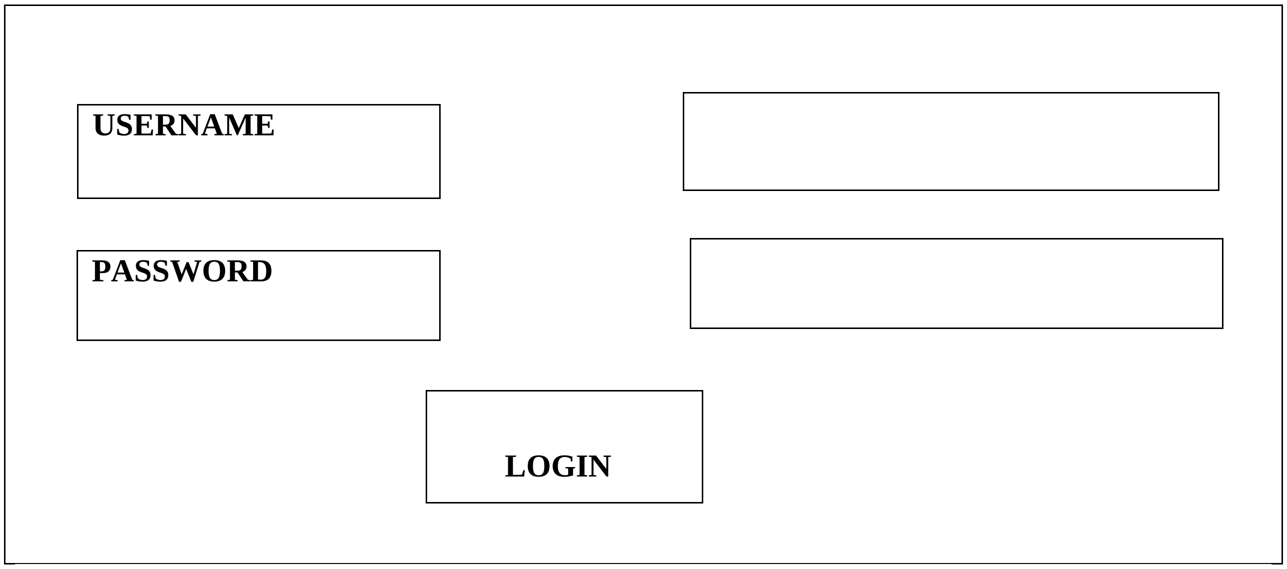
COLUMN NAME	DATATYPE	LENGHT	CONSTRAINT	DESCRIPTION
appointment_id	Int	10	Primary Key	Primary key of the table
User_id	Int	10	Not Null	User table’s primary key
Organization_id	Int	10	Not Null	Organization tables’s primary key
date	Date	10	Not Null	Appointment date
time	Time	10	Not Null	Appointment time
status	Varchar	20	Not Null	Appointment status

DAILY WORK

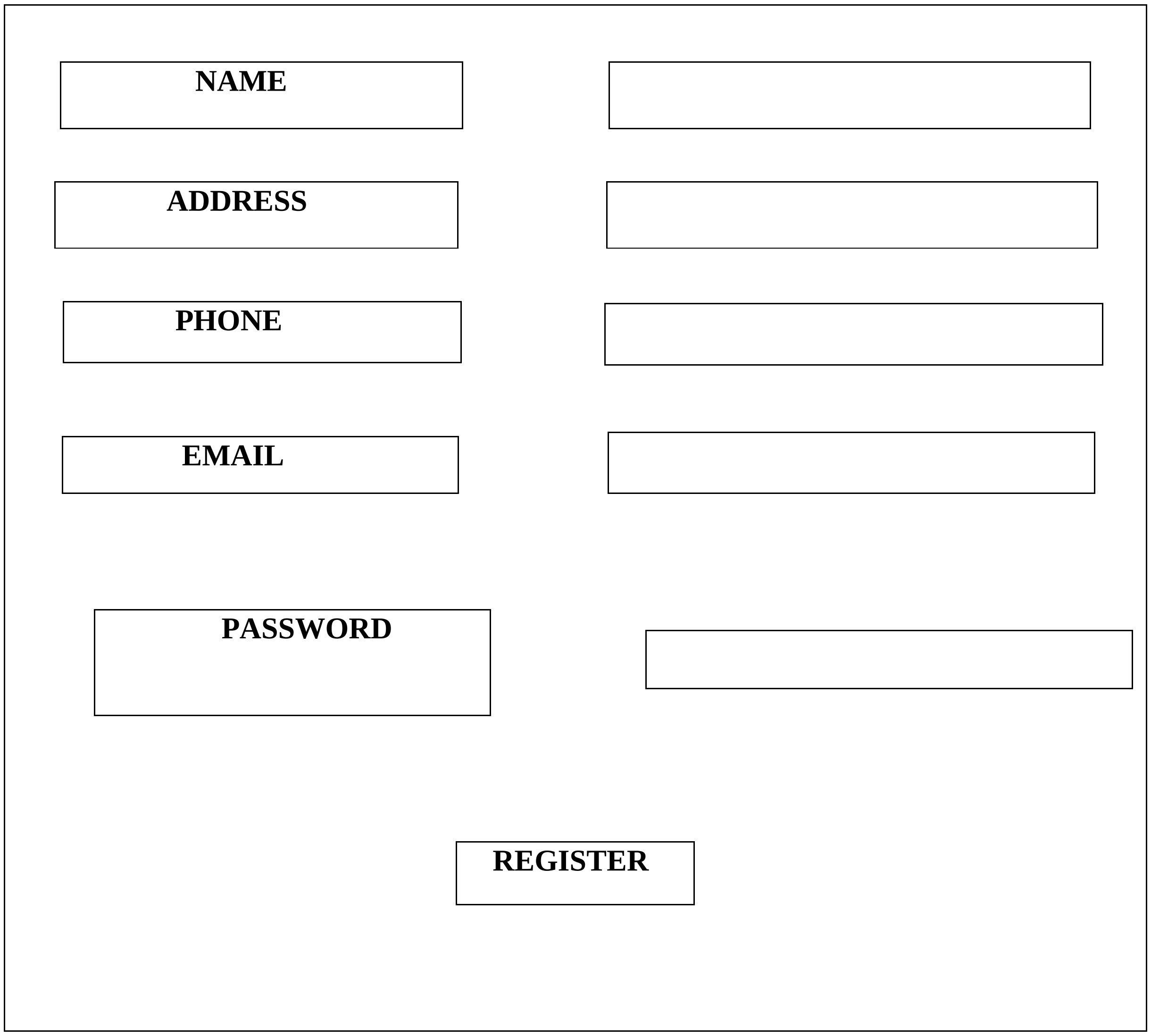
COLUMN NAME	DATATYPE	LENGHT	CONSTRAINT	DESCRIPTION
dw_id	Int	10	Primary Key	Primary key of the table
organization_id	Int	10	Foreign Key	Organization table’s primary key
work	Varchar	20	Not Null	Work Description
description	Varchar	50	Not Null	Description about the work
time	Time	10	Not Null	Daily work time

USER INTERFACE DESIGN

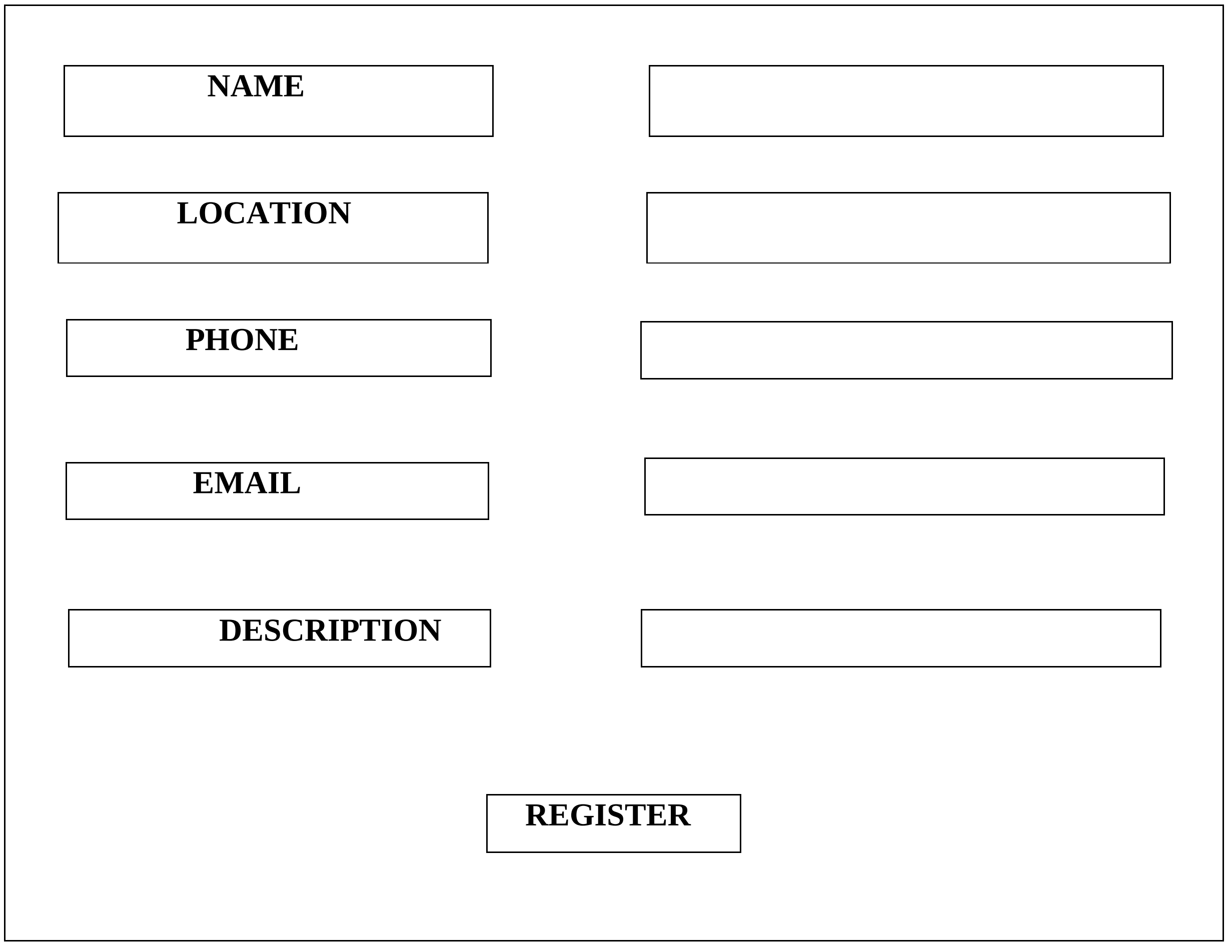
LOGIN



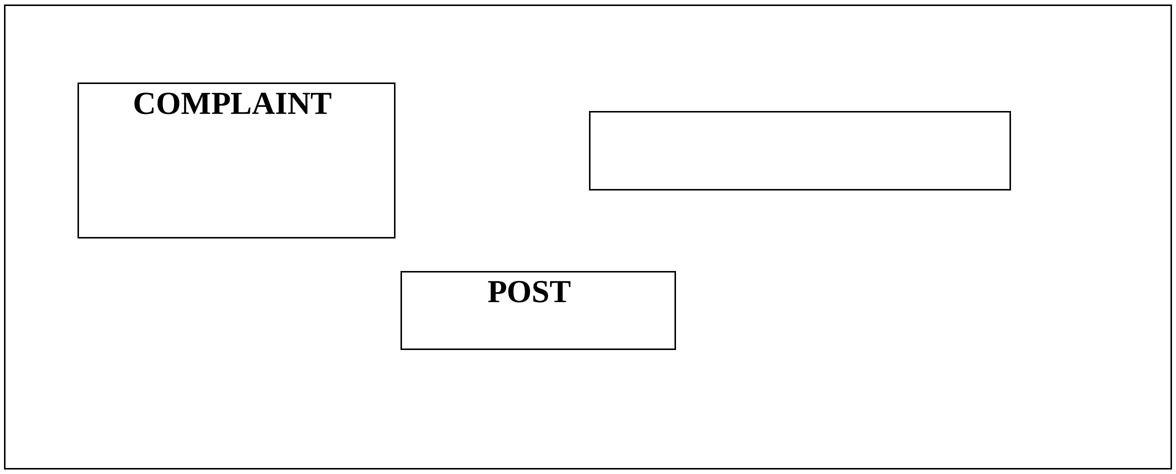
USER REGISTRATION



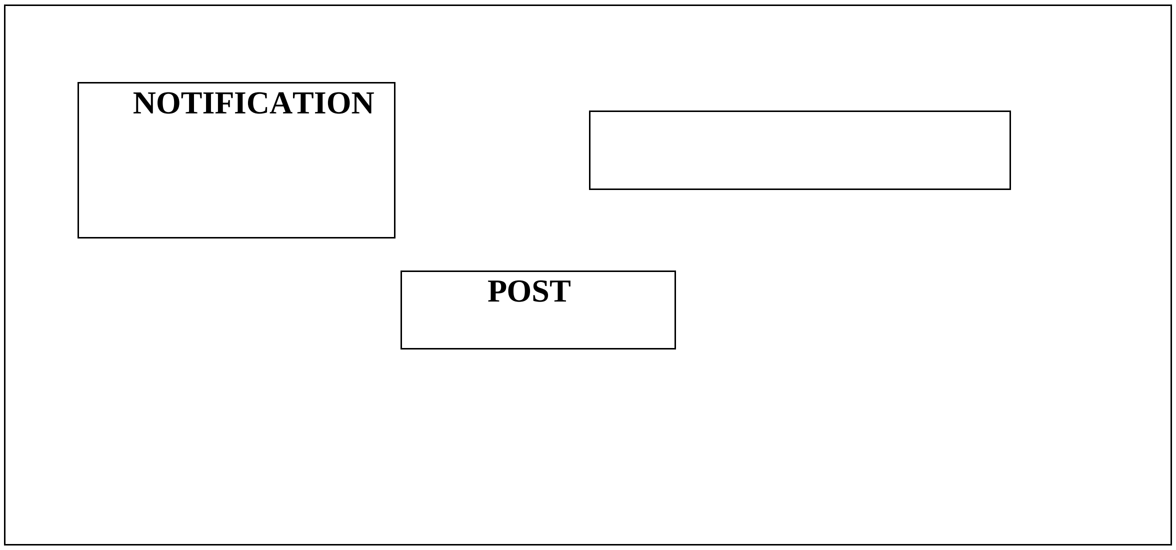
ORGANIZATION REGISTRATION



COMPLAINT



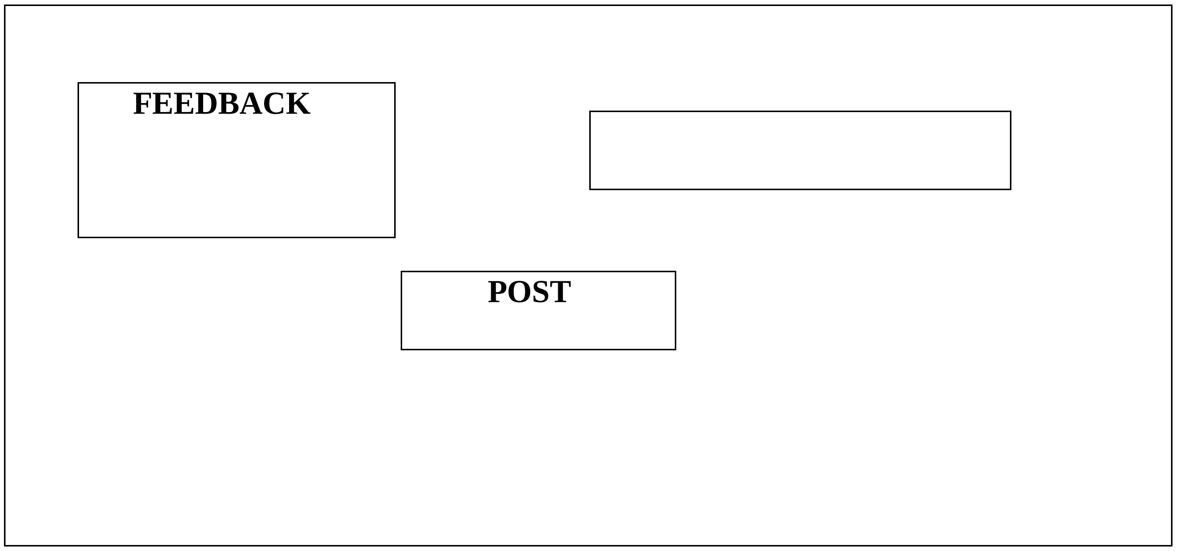
NOTIFICATION



VIEW COMPLAINT

COMPLAINT	USERID	DATE	TIME	
				REPLY

FEEDBACK



TEST CASES

A test case is a document, which has a set of test data, preconditions, expected results and post conditions, developed for a particular test scenario in order to verify compliance against a specific requirement. Test Case acts as the starting point for the test execution, and after applying a set of input values, the application has a definitive outcome and leaves the system at some end point or also known as execution post condition.

UNIT TEST

Unit testing is a Level of Testing where the smallest part of individual unit/component (called unit) is tested to determine if they are fit for use.

The unit test cases writing and execution is done by the developer (not the tester) to make sure that individual units are working as expected. The smallest part of individual components like functions, procedures, classes, interfaces, etc. If we take an example of functions when we pass input parameters to functions then check if the function should return an expected value. The main intention of this activity is to check whether units are working as per the design and handling error and exception more neatly. Both positive and negative conditions should handle properly.

The white box testing is used to test the unit testing. This is the very first step in the level of testing and started before doing integration testing.

UNIT TESTING TEST CASE PREPARATION GUIDELINES:

Unit Test Plan/Cases should be made a separate deliverable. It should not be merged with other artefacts. Try to document all the probable test scenarios which encompass uncommon and alternative flows. Once a project moves into the construction phase, the developers have a tendency to catch only the success situations or the situations which have been coded. Construction and Unit testing need to be made distinct phases and the deliverable needs to be scheduled accordingly. If Construction and UT are scheduled as a single phase, Unit testing results need to be made as a separate deliverable – This would help in filtering out any mistakes in the business flows at a nascent stage instead of in the integration testing or system testing which is more expensive.

Make use of the count of test cases planned, executed, passed, and failed to apprehend the progress and replicate Unit testing if required. Try to include on-the-fly test cases that are developed while executing a pre-defined set of test cases.

UNIT TESTING TEST CASES PREPARATION GUIDELINES CHECKLIST

INPUT DATA VALIDATION:

This section encompasses a range of checks that may be adapted generally to the data which is entered into an application system.

1. Mandatory Fields testing

2. Unique Field Values testing

3. Null value testing

4. The field accepts allowable characters only

5. Negative values testing

6. The field is limited to the field length specifications

7. Improbable Value testing

8. Garbage Value testing

9. Check for inter- Field dependencies

10. Equivalence Class Partitioning and boundary condition testing

11. Ensure error notification and correction processing is robust

THIS CONSTITUTES A SET OF CONDITIONS TO THE DATE FIELDS

1. Various Date Formats

2. US or UK Style Date Format

3. Positive testing – valid dates

4. Negative testing- invalid Dates like

5. A month does not accept 00 and 13 as values

6. Day do not contain 00 and 32 as its values

7. 28, 29, 30 are validated correctly.

8. Check for the impact of Weekends and Bank Holidays if applicable

9. The link between Leap Years and 29th February

SYSTEM INTERFACES:

This constitutes a set of conditions to the fields which are transferred between multiple application systems.

1. Check if all fields/parameters on an interface are exercised properly

2. All data fields need to work properly as per the validation List

3. Security testing across automated Interfaces

Check for the parent-child relationships

This constitutes a set of conditions that helps to verify the security of an application system.

1. Negative testing- Password is not visible

2. Access testing- multiple levels

3. Positive testing- Change Password

4. Error messages should not reveal any system information

5. Check if SSL is correctly deployed

6. Check if Lockout rules are applied

7. Check if the password saved in clear or encrypted?

8. Verify the application with valid Use rid and invalid user IDs

9. Verify the application with valid password and various invalid passwords

10. Check for access to the application by directly entering a valid URL(s). The system
Should ask for login details.

11. Ensure Browser(s) does not remember Passwords

SYSTEM TESTING

SYSTEM TESTING is a level of testing that validates the complete and fully integrated software product. The purpose of a system test is to evaluate the end-to-end system specifications. Usually, the software is only one element of a larger computer-based system. Ultimately, the software is interfaced with other software/hardware systems. System Testing is actually a series of different tests whose sole purpose is to exercise the full computer-based system.

In this tutorial, we will learn

- System Testing is Black box
- What do you verify in System Testing
- Software Testing Hierarchy
- Different Types of System Testing
- What Types of System Testing Should Testers Use

Two Category of Software Testing

- Black Box Testing

- White Box Testing

System test falls under the black box testing category of software testing. White box testing is the testing of the internal workings or code of a software application. In contrast, black box or System Testing is the opposite. System test involves the external workings of the software from the user's perspective.

Functions in System testing

Testing the fully integrated applications including external peripherals in order to check how components interact with one another and with the system as a whole. This is also called End to End testing scenario. Verify thorough testing of every input in the application to check for desired outputs. Testing of the user's experience with the application.

That is a very basic description of what is involved in system testing. You need to build detailed test cases and test suites that test each aspect of the application as seen from the outside without looking at the actual source code.

Different Types of System Testing

There are more than 50 types of System Testing. For an exhaustive list of software testing types click here. Below we have listed types of system testing a large software development company would typically use

- Usability Testing- mainly focuses on the user's ease to use the application, flexibility in
- Handling controls and ability of the system to meet its objectives
- Load Testing- is necessary to know that a software solution will perform under real-life
- Regression Testing- involves testing done to make sure none of the changes made over

- the course of the development process have caused new bugs. It also makes sure no old
- bugs appear from the addition of new software modules over time.
- Recovery testing - is done to demonstrate a software solution is reliable, trustworthy
- and can successfully recoup from possible crashes.
- Migration testing- is done to ensure that the software can be moved from older system
- Infrastructures to current system infrastructures without any issues.
- Functional Testing - Also known as functional completeness testing, Functional Testing involves trying to think of any possible missing functions. Testers might make a list of additional functionalities that a product could have to improve it during functional testing.
- Hardware/Software Testing - IBM refers to Hardware/Software testing as "HW/SW Testing". This is when the tester focuses his/her attention on the interactions between the hardware and software during system testing

CODING

The coding is the process of transforming the design of a system into a computer language format. This coding phase of software development is concerned with software translating design specification into the source code. It is necessary to write source code & internal documentation so that conformance of the code to its specification can be easily verified. Coding is done by the coder or programmers who are independent people than the designer. The goal is not to reduce the effort and cost of the coding phase, but to cut to the cost of a later stage. The cost of testing and maintenance can be significantly reduced with efficient coding.

COMPLETE PROJECT CODING

LOGIN

```
def login(request):
    if request.method=='POST':
        username=request.POST.get('un')
        password=request.POST.get('pws')
        obj=Login.objects.filter(username=username,password=password)
        print(len(obj))
        tp=""
        for ob in obj:
            tp=ob.type
            uid=ob.u_id
            if tp=="admin":
                request.session["u_id"]=uid
                return HttpResponseRedirect('/temp/admin/')
            elif tp=="organization":
                request.session["u_id"]=uid
                return HttpResponseRedirect('/temp/organization/')
            elif tp=="user":
                request.session["u_id"]=uid
                return HttpResponseRedirect('/temp/user/')
        else:
            objlist="username or password incorrect...please try again...."
            context={
                'msg':objlist
            }
            return render(request,'login/login.html',context)
    return render(request,'login/login.html')
```

REGISTER USER

```
def register(request):
    if request.method=='POST':
        obj=User()
        obj.name=request.POST.get('name')
        obj.address=request.POST.get('address')
        obj.phone=request.POST.get('phone')
        obj.email=request.POST.get('email')
        obj.username=request.POST.get('username')
        obj.password=request.POST.get('password')
        obj.save()


        ob = Login()
        ob.username = obj.username
        ob.password = obj.password
        ob.u_id = obj.user_id
        ob.type = 'user'
        ob.save()
    return render(request,'user/register.html')
```

REGISTER ORGANIZATION

```
def register(request):
    if request.method=='POST':
        obj=Organization()
        obj.name=request.POST.get('name')
        obj.location=request.POST.get('location')
        obj.phone=request.POST.get('phone')
        obj.email=request.POST.get('email')
        obj.username=request.POST.get('username')
        obj.password=request.POST.get('password')
        obj.description=request.POST.get('description')
        obj.status='pending'
        obj.save()
        ob = Login()

        ob.username = obj.username

        ob.password = obj.password

        ob.u_id = obj.organization_id

        ob.type = 'organization'

        ob.save()

    return render(request,'organization/register.html')
```

POST COMPLAINT

```
def post_complaint(request):

    uid = request.session["u_id"]

    if request.method=='POST':

        obj=Complaint()

        obj.complaint=request.POST.get('complaint')

        obj.date=datetime.datetime.today()

        obj.time=datetime.datetime.now()

        obj.reply='pending'

        obj.user_id=uid

        obj.save()

    return render(request,'complaint/post.html')
```

VIEW COMPLAINT

```
def view_complaint(request):
    obj=Complaint.objects.all()
    context={
        'a':obj
    }
    return render(request,'complaint/view_complaint.html',context)
```

POST REPLY

```
def post_reply(request,idd):
    if request.method=='POST':
        obj=Complaint.objects.get(c_id=idd)
        obj.reply=request.POST.get('reply')
        obj.save()
        return view_complaint(request)
    return render(request,'complaint/post_reply.html')
```

BOOKING APPOINMENT

```
def book(request):
    ob=Organization.objects.all()
    context={
        'a':ob
    }
    uid = request.session["u_id"]
    if request.method=='POST':
        obj=Appoinment()
        obj.organization_id=request.POST.get('o')
        obj.date=datetime.datetime.today()
        obj.time=datetime.datetime.now()
        obj.status='pending'
        obj.user_id=uid
        obj.save()
    return render(request, 'appoinment/book.html',context)
```

MANAGE BOOKING

```
def view_book(request):
    obj=Appoinment.objects.all()
    context={
        'a':obj
    }
    return render(request, 'appoinment/view_book.html', context)


def accept(request,idd):
    obj=Appoinment.objects.get(ap_id=idd)
    obj.status="accepted"
    obj.save()
    return view_book(request)
```

```
def ignore(request,idd):
    obj=Appoinment.objects.get(ap_id=idd)
    obj.status="ignored"
    obj.save()
    return view_book(request)
```

ADD NOTIFICATION

```
def post(request):
    if request.method=='POST':
        obj=Notification()
        obj.notification=request.POST.get('notification')
        obj.date=datetime.datetime.today()
        obj.time=datetime.datetime.now()
        obj.save()
    return render(request, 'notification/post.html')
```

DESCRIPTION OF CODING

- To translate the design of system into a computer language format: The coding is the process of transforming the design of a system into a computer language format, which can be executed by a computer and that perform tasks as specified by the design of operation during the design phase.
- To reduce the cost of later phases: The cost of testing and maintenance can be significantly reduced with efficient coding.

- Making the program more readable: Program should be easy to read and understand. It increases code understanding having readability and understand ability as a clear objective of the coding activity can itself help in producing more maintainable software.

CODE EFFICENCY

Code efficiency is a broad term used to depict the reliability, speed and programming methodology used in developing codes for an application. General way to achieve code efficiency: To remove unnecessary code or code that goes to redundant processing. To make use of optimal memory and non-volatile storage. General way to achieve code efficiency:

- To remove unnecessary code or code that goes to redundant processing
- To make use of optimal memory and nonvolatile storage
- To ensure the best speed or run time for completing the algorithm
- To make use of reusable components wherever possible
- To make use of error and exception handling at all layers of software, such as the user interface, logic and data flow
- To create programming code that ensures data integrity and consistency
- To develop programming code that's compliant with the design logic and flow. To make use of coding practices applicable to the related software
- To optimize the use of data access and data management practices

STANDARDISATION OF CODE

Coding standards are a set of guidelines, best practices, programming styles and conventions that developers adhere to when writing a piece of code for a project. All the big software companies have them. And advise their developers to embed these coding standards in deliverables. Especially before deployment in a production environment

```
def add(request,idd):
```

```
tid = request.session["uid"]

cl = request.session["cc"]

 if request.method == "POST":

ob = Extra() ob.date = datetime.datetime.today()

 ob.std_id = idd ob.t_id = tid

ob.cls = cl ob.activity = request.POST.get('act')

 ob.save()

return eview(request)

return render(request,'extra/add.html')
```

ERROR HANDLING

Anything between the BEGIN TRY and END TRY is the code that we want to monitor for an error. So, if an error would have happened inside this TRY statement, the control would have immediately get transferred to the CATCH statement and then it would have started executing code line by line:

```
def update_teacher(request,idd):

 try: obj = teacher.objects.filter(u_id=idd)

context = { 'objval': obj, }

 if request.method == "POST" and 'update' in request.POST:

 obj = User.objects.get(u_id=idd) obj.name = request.POST.get('name')

 obj.address = request.POST.get('addr')

obj.phone = request.POST.get('phone')
obj.gender = request.POST.get('gender')

obj.email = request.POST.get('email')

obj.password = request.POST.get('pass')

obj.save()

return HttpResponseRedirect('/user/manage_teacher/')

 except: context={ 'msg': 'not updated' }

 return render(request,'teacher/update_teacher.html',context)
```

PARAMET CALLING/PARSE

A value-type variable contains its data directly as opposed to a reference-type variable, which contains a reference to its data. Passing a value-type variable to a method by value means passing a copy of the variable to the method. Any changes to the parameter that take place inside the method have no effect on the original data stored in the argument variable. If you want the called method to change the value of the argument, you must pass it by reference, using the ref or out keyword. You may also use the in keyword to pass a value parameter by reference to avoid the copy while guaranteeing that the value will not be changed.

```
class at_view(APIView):

 def get(self,request):

ob=Attendence.objects.all()

 ser=android(ob,many=True)

 return Response(ser.data)

 obj.save()

return HttpResponse('okk')
```

VALIDATION CHECKS

whenever the user accepts an input, it needs to be checked for validation which checks if the input data is what we are expecting. The validation can be done in two different ways. That is using a flag variable or buy using try or except which the flag variable will be set to false initially and if we can find out that the input data is what we are expecting the flag status can be set to true and find out what can be done next based on the status of the flag whereas while using try or except, a section of code is tried to run.

TESTING

Testing is an activity to verify that a correct system is being built and is performed with the intent of finding faults in the system. However not restricted to being performed after the development phase is complete, but this is to carry out in parallel with all stages of system development, starting with requirements specification. Testing results, once gathered and evaluated, provide a qualitative indication of software quality and reliability and serve as a basis for design modification if required. A project is said to be incomplete without proper testing.

System testing is a process of checking whether the developed system is working according to the original objectives and requirements. The system should be tested experimentally with test data so as to ensure that system works according to the required specification. When the system is found working, test it with actual data and check performance.

TESTING PRINCIPLES

All tests should be traceable to customer requirements. The focus of testing will shift progressively from programs. Exhaustive testing is not possible. To be more effective, testing should be one, which has probability of finding errors.

The following are attributes of good test:

- A good test has a high probability of finding errors.
- A good test is not redundant.
- A good test should be ―best of bree.
- A good test should neither too simple nor too complex

TEST USED

The details of the software functionality tests are given below. The testing procedure that has been used as follows:

- Unit Testing
- Integration Testing
- Validation Testing
- Output Testing
- User Acceptance Testing

UNIT TESTING

The first level of testing is called as unit testing. Here the different modules are tested and the specification produced during design for the modules. Unit testing is essential for verification of the goal and to test the internal logic of the modules. Unit testing is conducted to different modules of the project. Errors were noted down and corrected down immediately and the program clarity was increased. The testing was carried out during the programming stage itself. In this step each module is found to be working satisfactory as regard to be expected out from the module.   For example, the Login page is tested against three different states that are a positive input, a negative input and a 0 input. Testing with a positive input, and with a negative input will behave as expected. Testing with a 0 input however will yield a surprise. This is just one example of why it makes sense to focus on testing the different states of your code

INTEGRATION TESTING

The second level of testing includes integration testing. It is a systematic testing of constructing structure. At the same time tests are conducted to uncover errors with the interface. It need not to be the case, that software whose modules when run individually showing results will also show perfect results when run as a whole. The individual modules are tested again and the results are verified. The goal is to see if the modules integrated between the modules. This testing activity can be considered as testing the design and emphasizes on testing modules interaction.

VALIDATION TESTING

The next level of testing is validation testing. Here the entire software is tested. The reference document for this process is the requirement and the goal is to see if the software meets its requirements. The requirement document reflects and determines whether the software functions as the user expected. At culmination of integration testing, software is completely assembled as a package and corrected and a final series of software test validation test begins. The proposed system under construction has been tested by using validation testing and found to be working satisfactory. Data validation checking is done to see whether the corresponding entries made in different tables are done correctly. Proper validation checks are done in case of insertion and updatingof tables, in order to see that no duplication of data has occurred. If any such case arises proper warning message will be displayed. Double configuration is done before the administrator deletes a data in order to get positive results and to see that o data have been deleted by accident. Story narrator is a website done using ASP.NET 3 tier architecture, which includes 3 tiers in its design.

Presentation Layer - ASP.NET Web Form project that contains .aspx, .ascx and other types of files that is helpful to create a User Interface (UI) of the application or website or any other type of application. 2. Business Access Layer (BAL) - This tier contains business logic, secondary validations, calculations and call to Data Access Layer methods if needed. If there is no business logic and any other types of validations to be performed, it simply calls Data Access Layer on need basis. 3. Data Access Layer (DAL) - This tier contains necessary code to connect to the database and perform database activities (such as insert, update, delete, and load records) and returns result to the Business Access Layer.

OUTPUT TESTING

The output of the software should be acceptable to the system user. The output of requirement is defined during the system analysis. Testing of the software system is done against the output and the output testing was completed with success.

USER ACCEPTANCE TESTING

An acceptance test has the objective of selling the user on the validity and reliability of the system. It verifies that the system procedures operate to system specification and the integrity of the vital data is maintained.

BLACK BOX TESTING

Black box testing, also called behavioural testing, focuses on the functional requirements of the software. That is black box testing techniques enable you to derive sets of input conditions that will fully exercise all functional requirements for a program

TEST PLAN

A test plan documents the what, when, why, how, and who of a testing project. It lays out the overall objective and scope of the tests to be run. The purpose of a test plan is to set expectations and align the team for success throughout the entire testing process.

A test plan does not include the individual test cases, as it is meant to be a higher-level document. A test plan ultimately aligns teams to deliver a more reliable product to their users by orchestrating a smooth testing process.

Test planning is the first thing that should happen in the software testing life cycle. The test plan document is a living and breathing artefact – it is dynamic in the sense that it should always be up to date. So, when plans change, the test plan should be updated in order to maintain an accurate set of information for the team

TYPES OF TEST PLANS IN SOFTWARE TESTING

Test plans can be used as supporting documentation for an overall testing objective (a master test plan) and specific types of tests (a testing type specific plan).

Master test plan: A master test plan is a high-level document for a project or product's overall testing goals and objectives. It lists tasks, milestones, and outlines the size and scope of a testing project. It encapsulates all other test plans within the project.

Testing type specific plan: Test plans can also be used to outline details related to a specific type of test. For example, you may have a test plan for unit testing, acceptance testing, and integration testing. These test plans drill deeper into the specific type of test being conducted

UNIT AND SYSTEM TEST REPORT

Test Report is a document which contains a summary of all test activities and final test results of a testing project. Test report is an assessment of how well the Testing is Performed Based on the test report, stakeholders can evaluate the quality of the tested product and make a decision on the software release. For example, if the test report informs that there are many defects remaining in the product, stakeholders can delay the release until all the defects are fixed. It is very common that a CI/CD pipeline contains a test job that will verify your code. If the tests fail, the pipeline fails and users get notified. The person that works on the merge request will have to check the job logs and see where the tests failed so that they can fix them. You can configure your job to use Unit test reports, and GitLab will display a report on the merge request so that it's easier and faster to identify the failure without having to check the entire log. Unit test reports currently only support test reports in the JUnit report format. If you don't use Merge Requests but still want to see the unit test report output without searching through job logs, the full Unit test reports are available in the pipeline detail view. Consider the following workflow:

Your master branch is rock solid, your project is using GitLab CI/CD and your pipelines indicate that there isn't anything broken. 2. Someone from your team submits a merge request, a test fails and the pipeline gets the known red icon. To investigate more, you have to go through the job logs to figure out the cause of the failed test, which usually contain thousands of lines. 3. You configure the Unit test reports and immediately GitLab collects and exposes them in the merge request. No more searching in the job logs. 4. Your development and debugging workflow becomes easier, faster and efficient.

How it works

First, GitLab Runner uploads all JUnit report format XML files as artifacts to GitLab. Then, when you visit a merge request, GitLab starts comparing the head and base branch's JUnit report format XML files, where:

- The base branch is the target branch (usually master).
- The head branch is the source branch (the latest pipeline in each merge request).

The reports panel has a summary showing how many tests failed, how many had errors and how many were fixed. If no comparison can be done because data for the base branch is not available, the panel will just show the list of failed tests for head. There are four types of results:

- Newly failed tests: Test cases which passed on base branch and failed on head branch
- Newly encountered errors: Test cases which passed on base branch and failed due to a test error on head branch
- Existing failures: Test cases which failed on base branch and failed on head branch
- Resolved failures: Test cases which failed on base branch and passed on head branch

- Each entry in the panel will show the test name and its type from the list above. Clicking on the test name will open a modal window with details of its execution time and the error output.

DEBUGGING AND CODE IMPROVEMENT

In ideal worlds, all programmers would be so skilled and attentive to detail that they would write bug-free code. Unfortunately, we do not live in an ideal world. As such, debugging, or tracking down the source of errors and erroneous result, is an important task that all developers need to perform before they allow end-user to use their applications. We will discuss some techniques for reducing the number of bugs in code up front.

There are three categories of bugs

SYNTAX ERROR:

These errors occur when code breaks the rule of the language, such as visual Basic sub statement without a closing End sub, or a forgotten closing curly braces ({}) in c#. Theses error the easiest to locate. The language complier or integrated development environment (IDE) will alert you to them and will not allow you to compile your program until you correct them.

SEMANTIC ERROR

These errors occur in code that is correct according to rules of the compiler, but that causes unexpected problems such as crashes or hanging on execution. A good example is code that execute in a loop but never exists the loop, either because the loop depends on the variable whose values was expected to be something different than it actually was or because the programmer forget to increment the loop counter. Another category of errors in this area includes requesting a field from a dataset, there is no way to tell if the field actually exists at compile time. these bugs are harder to detect and are one type of running error.

LOGIC ERROR

Logic errors are like semantic errors, logic errors are runtime error. That is, they occur while the program is running. But unlike semantic errors, logic errors do not cause the application to crash or hang. Logic error results in unexpected values or output. This can be a result of something as simple as a mistyped variables name that happens to match another declared variable in the program. This type of error can be extremely difficult to track down to eliminate.

SYSTEM SECURITY MESSURES

DATABASE/DATA SECURITY

DATABASE SECURITY

Database security encompasses a range of security controls designed to protect the Database Management System (DBMS). The types of database security measures your business should use include protecting the underlying infrastructure that houses the database such as the network and servers), securely configuring the DBMS, and the access to the data itself.

DATABASE SECURITY CONTROLS

Database security encompasses multiple controls, including system hardening, access, DBMS configuration, and security monitoring. These different security controls help to manage the circumventing of security protocols.

SYSTEM HARDENING AND MONITORING

The underlying architecture provides additional access to the DBMS. It is vital that all systems are patched consistently, hardened using known security configuration standards, and monitored for access, including insider threats.

DBMS CONFIGURATION

It is critical that the DBMS be properly configured and hardened to take advantage of security features and limit privileged access that may cause a misconfiguration of expected security settings. Monitoring the DBMS configuration and ensuring proper change control processes helps ensure that the configuration stays consistent.

AUTHENTICATION

Database security measures include authentication, the process of verifying if a user’s credentials match those stored in your database, and permitting only authenticated users access to your data, networks, and database platform.

ACCESS

A primary outcome of database security is the effective limitation of access to your data. Access controls authenticate legitimate users and applications, limiting what they can access in your database. Access includes designing and granting appropriate user attributes and roles and limiting administrative privileges.

DATABASE AUDITING

Monitoring (or auditing) actions as part of a database security protocol delivers centralized oversight of your database. Auditing helps to detect, deter, and reduce the overall impact of unauthorized access to your DBMS.

BACKUPS

A data backup, as part of your database security protocol, makes a copy of your data and stores it on a separate system. This backup allows you to recover lost data that may result from hardware failures, data corruption, theft, hacking, or natural disasters.

ENCRYPTION

Database security can include the secure management of encryption keys, protection of the encryption system, management of a secure, off-site encryption backup, and access restriction protocols.

APPLICATION SECURITY

Database and application security framework measures can help protect against common known attacker exploits that can circumvent access controls, including SQL injection.

Why is database security important? Safeguarding the data your company collects and manages is of utmost importance. Database security can guard against a compromise of your database, which can lead to financial loss, reputation damage, consumer confidence disintegration, brand erosion, and non-compliance of government and industry regulation.

Database security safeguards defend against a myriad of security threats and can help protect your enterprise from:

- Deployment failure
- Excessive privileges
- Privilege abuse
- Platform vulnerabilities
- Unmanaged sensitive data
- Backup data exposure
- Weak authentication
- Database injection attacks

Database security safeguards defend against a myriad of security threats and can help protect your enterprise from:

- Deployment failure
- Excessive privileges
- Privilege abuse
- Platform vulnerabilities
- Unmanaged sensitive data
- Backup data exposure
- Weak authentication
- Database injection attacks

COST ESTIMATION

The other cost of requirements is the amount of money or effort that you have to spend building them into a product. Once the requirements specification is complete, you can use one of the estimating methods to assess the cost, expressing the result as a monetary amount or time to build. There is no best method to use when estimating. Keep in mind, however, that your estimates should be based on some tangible, countable artifact. If you are using this template, then, as a result of doing the work of requirements specification, you are producing many measurable deliverables.

For example:

● Number of input and output flows on the work context

● Number of business events

● Number of product use cases

● Number of functional requirements

● Number of non-functional requirements

● Number of requirements constraints

● Number of function points

The more detailed the work you do on your requirements, the more accurate your deliverables will be. Your cost estimate is the amount of resources you estimate each type of deliverable will take to produce within your environment. You can create some very early cost estimates based on the work context. At that stage, your knowledge of the work will be general, and you should reflect this vagueness by making the cost estimate a range rather than a single figure. As you increase your knowledge of the requirements, we suggest you try using function point counting—not because it is an inherently superior method, but because it is so widely accepted.

So much is known about function point counting that it is possible to make easy comparisons with other products and other installations' productivity. It is important that your client be told at this stage what the product is likely to cost. You usually express this amount as the total cost to complete the product, but you may also find it advantageous to point out the cost of the requirements effort, or the costs of individual requirements. 76 Whatever you do, do not leave the costs in the lap of hysterical optimism. Make sure that this section includes meaningful numbers based on tangible deliverables.

EMPIRICAL ESTIMATION TECHNIQUE

Empirical estimation is a technique or model in which empirically derived formulas are used for predicting the data that are a required and essential part of the software project planning step. These techniques are usually based on the data that is collected previously from a project and also based on some guesses, prior experience with the development of similar types of projects, and assumptions. It uses the size of the software to estimate the effort. In this technique, an educated guess of project parameters is made. Hence, these models are based on common sense. However, as there are many activities involved in empirical estimation techniques, this technique is formalized. For example Delphi technique and Expert Judgement technique.

HEURISTIC TECHNIQUE

Heuristic word is derived from a Greek word that means "to discover". The heuristic technique is a technique or model that is used for solving problems, learning, or discovery in the practical methods which are used for achieving immediate goals. These techniques are flexible and simple for taking quick decisions through shortcuts and good enough calculations, most probably when working with complex data. But the decisions that are made using this technique are necessary to be optimal. In this technique, the relationship among different project parameters is expressed using mathematical equations. The popular heuristic technique is given by Constructive Cost Model (COCOMO). This technique is also used to increase or speed up the analysis and investment decisions.

ANALYTICAL ESTIMATION TECHNIQUE

Analytical estimation is a type of technique that is used to measure work. In this technique, firstly the task is divided or broken down into its basic component operations or elements for analysing. Second, if the standard time is available from some other source, then these sources are applied to each element or component of work. Third, if there is no such time available, then the work is estimated based on the experience of the work. In this technique, results are derived by making certain basic assumptions about the project. Hence, the analytical estimation technique has some scientific basis. Halstead’s software science is based on an analytical estimation model.

REPORTS

INPUT AND OUTPUT DESIGN

In an information system, input is the raw data that is processed to produce output. During the input design, the developers must consider the input devices such as PC, MICR, OMR, etc. Therefore, the quality of system input determines the quality of system output. Well-designed input forms and screens have following properties

- It should serve specific purpose effectively such as storing, recording, and retrieving the information.
- It ensures proper completion with accuracy.
- It should be easy to fill and straightforward.
- It should focus on user’s attention, consistency, and simplicity.
- All these objectives are obtained using the knowledge of basic design principles regarding

➢ What are the inputs needed for the system?

➢ How end users respond to different elements of forms and screens?

The design of output is the most important task of any system. During output design, developers identify the type of outputs needed, and consider the necessary output controls and prototype report layouts.

OBJECTIVES OF OUTPUT DESIGN

The objectives of input design are:

- To develop output design that serves the intended purpose and eliminates the production of unwanted output.

- To develop the output design that meets the end users requirements.

- To deliver the appropriate quantity of output.

- To form the output in appropriate format and direct it to the right person.

- To make the output available on time for making good decisions.

INPUT DESIGN

The first step in system design is to design input and output within predefined guidelines. In input design, user originated inputs are converted into computer based format. In output design, the emphasis is on producing the hard copy of the information requested or displaying the output on a CRT/LCD screen in a predefined format. The following features have been incorporated into the input design of the proposed system. In this system, Input design consists of developing specifications and procedures for data preparation, those steps necessary to put the input into desired output. Inaccurate input data are the most common causes of error in the system processing. Proper messages and appropriate direction can control errors committed by users.
Easy Data Input

Data entry has been designed in a manner much similar to the source documents. Appropriate messages are provided in the message area, which prompts the user in entering the right data. Erroneous data inputs are checked at the end of each screen entry.

Data Validation

The input data is validated to minimize errors in data entry. For certain data specific codes have been given and validation is done which enables the user to enter the required data or correct them if they entered wrong codes. It uses both server side and client side validations.

USER FRIENDLINESS

User is never left in a state of confusion as to what is happening, instead appropriate error and acknowledge messages are sent. Error maps are used to indicate the error codes and specific error messages. Message dialogue boxes showing for successful operations by using Script Manager and Link buttons are used for redirecting, etc.

CONSISTENT FORMAT

A fixed format is adopted for displaying the title messages. Every screen has line, which displays the operation that can be performed after the data entry. They are normally done at the touch of a key.

INTERACTIVE DIALOGUE

The system engages the user in an interactive dialogue. The system is able to extract missing or omitted information from the user by directing the user through appropriate messages, which are displayed.

OUTPUT DESIGN

The output is the most important and direct source of information to the user. The output should be provided in a most efficient formatted way. Based on the options given by the users and the administrator various types of output screens have been generated.

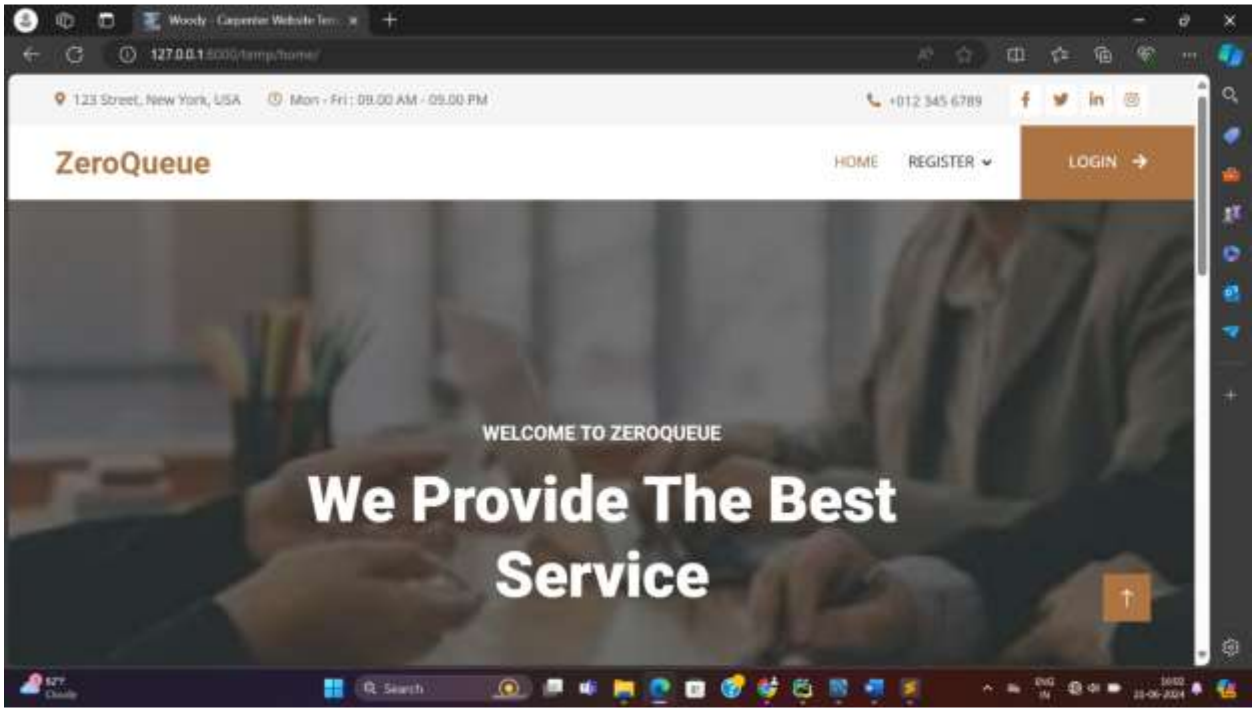
The computer output is the most important and direct source of information to the user. Efficient and intelligible output design improves the system‘s relationship with the user and helps in decision-making. Output design was studied going actively during the study phase. The objective of the output design is defined the contents and format of all documents and reports in an attractive and useful format.

Types of Outputs

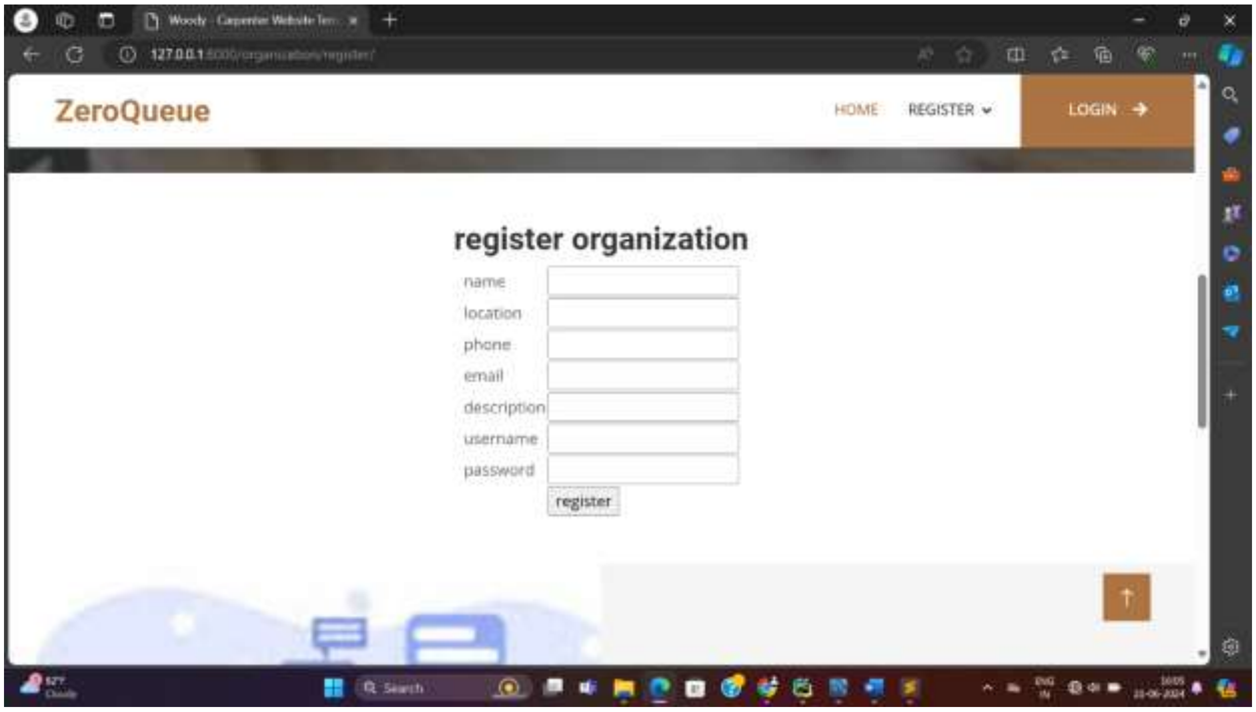
- External outputs
- Internal outputs
- Operational outputs
- Interactive outputs

Output generally refers to the results and information that are generated by the system. It can be in the form of operational documents and reports. The major form of output is a hard copy from the printer. Once the output requirements are identified, the output devices used also should be determined. Factors like compatibility of the output device with the system and response time requirement should be considered while descending the output device to be utilized.

HOME


123 Street, New York, USA
Mon - Fri : 09.00 AM - 09.00 PM
+012 345 6789
ZeroQueue
HOME
REGISTER
LOGIN
WELCOME TO ZEROQUEUE
We Provide The Best Service

ORGANIZATION REGISTER

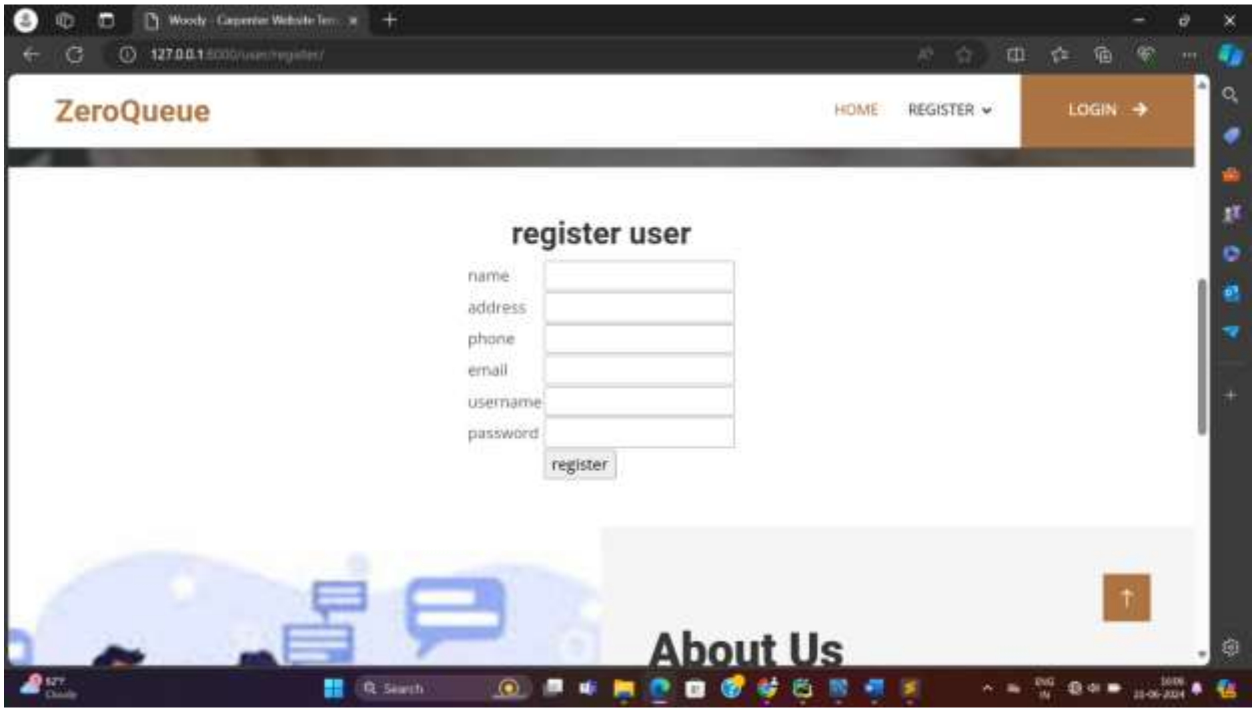


register organization

name	
location	
phone	
email	
description	
username	
password	

register

USER REGISTER

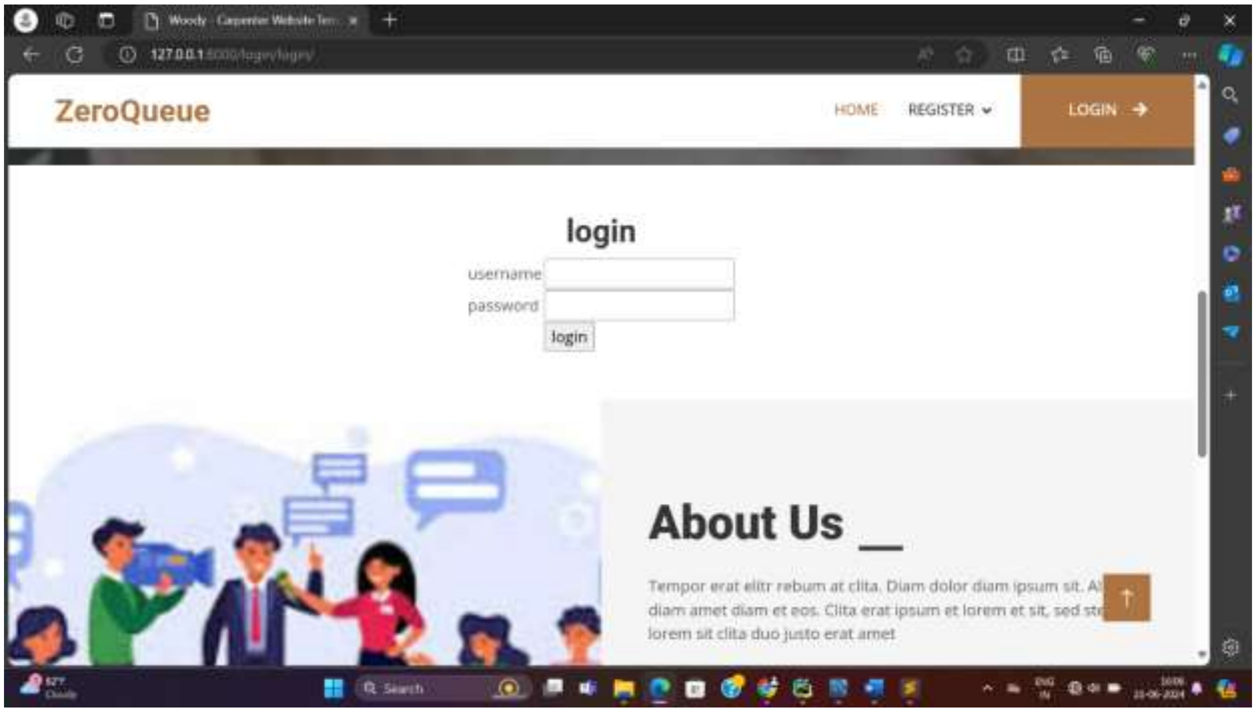


register user

name	
address	
phone	
email	
username	
password	

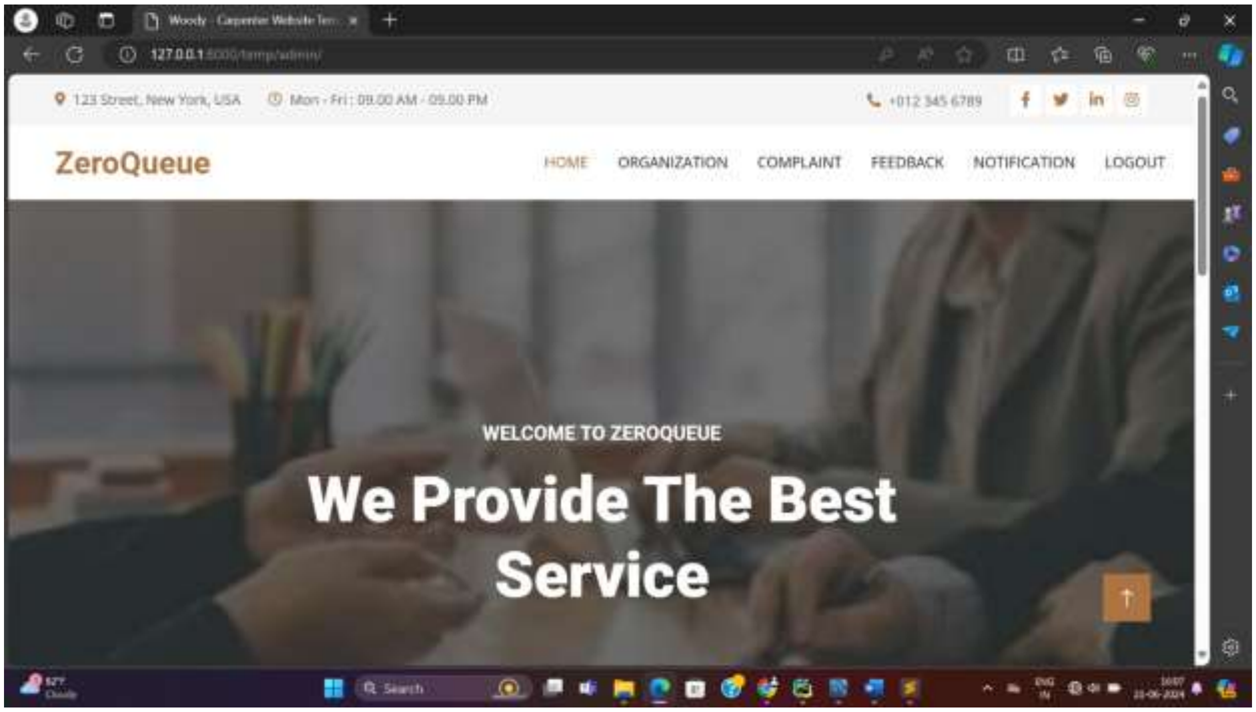
register

LOGIN

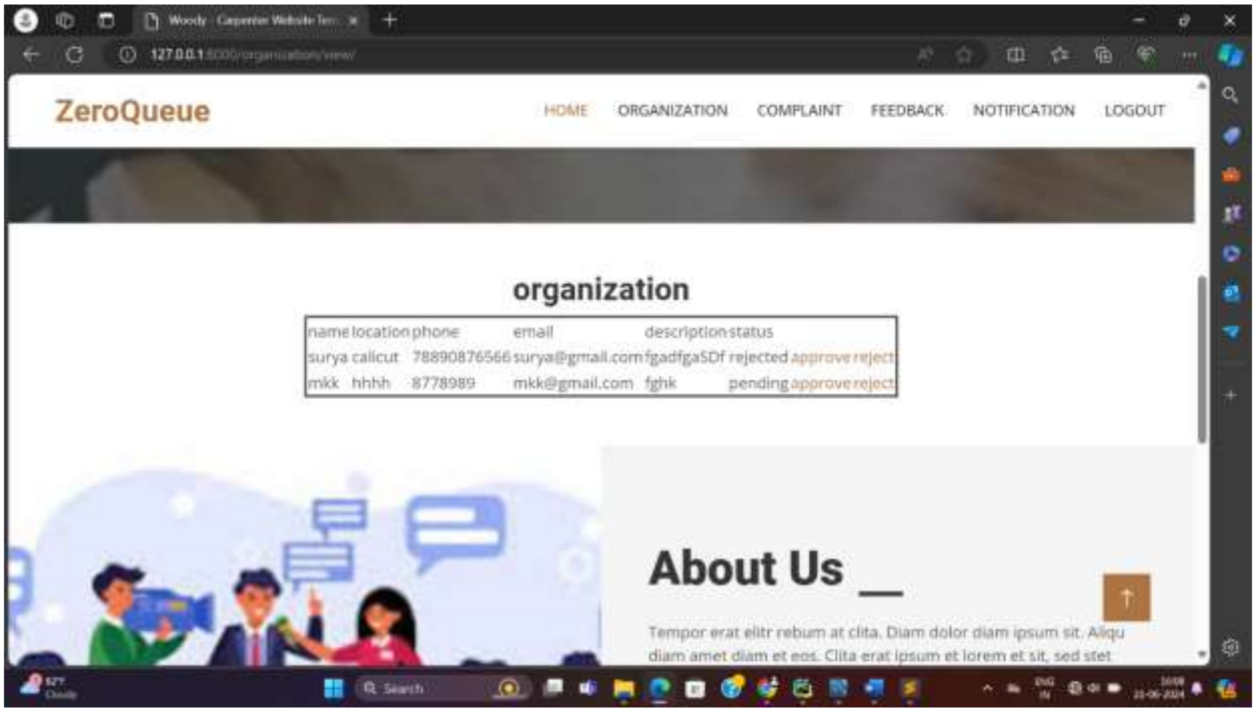


Tempor erat elitr rebum at clita. Diam dolor diam ipsum sit. Al diam amet diam et eos. Clita erat ipsum et lorem et sit, sed ste lorem sit clita duo justo erat amet

ADMIN



ORGANIZATION MANAGE



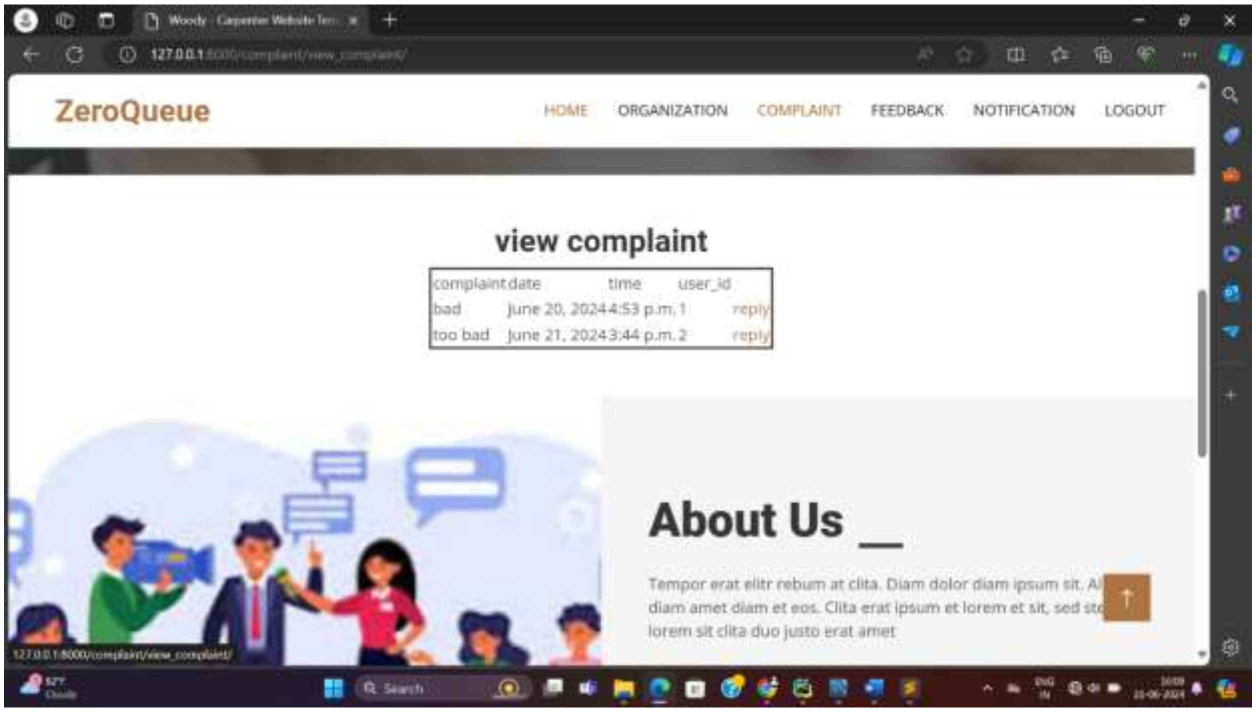
organization

name	location	phone	email	description	status		
surya	calicut	78890876566	surya@gmail.com	fgadfgaSDf	rejected	approve	reject
mkk	hhhh	8778989	mkk@gmail.com	fghk	pending	approve	reject

About Us

Tempor erat elitr rebum at clita. Diam dolor diam ipsum sit. Aliqu diam amet diam et eos. Clita erat ipsum et lorem et sit, sed stet

VIEW COMPLAINT



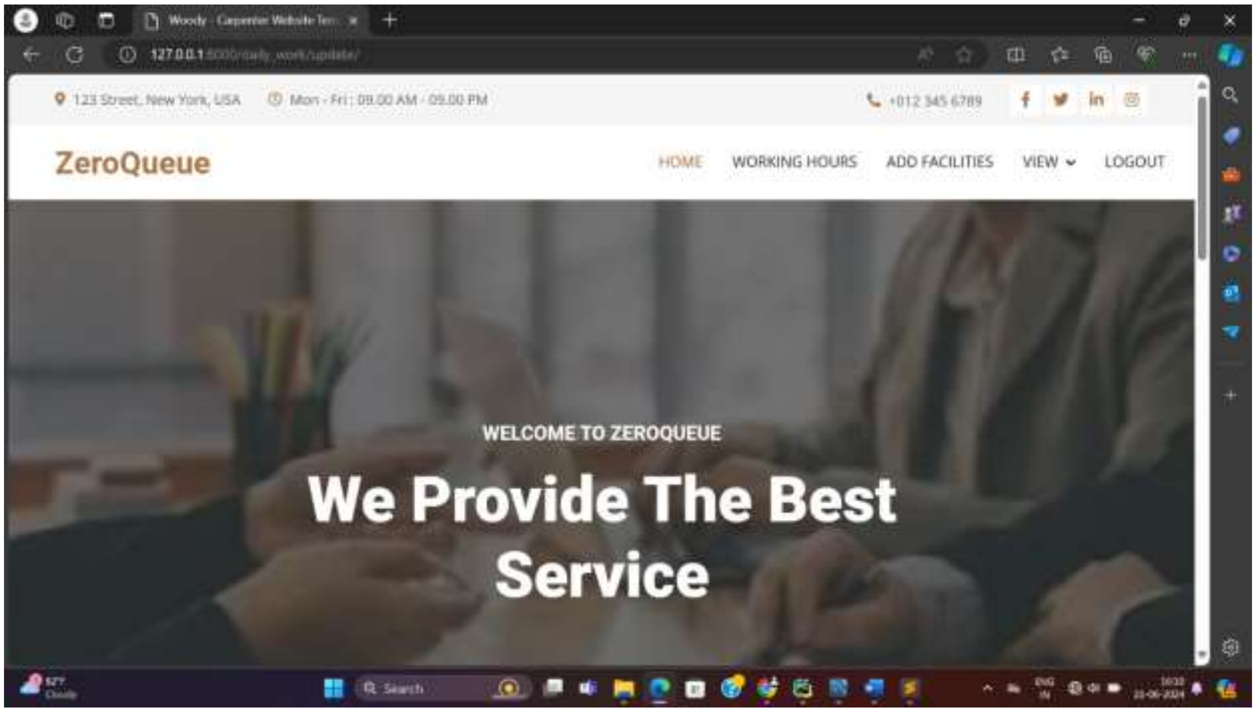
view complaint

complaint	date	time	user_id	
bad	June 20, 2024	4:53 p.m.	1	reply
too bad	June 21, 2024	3:44 p.m.	2	reply

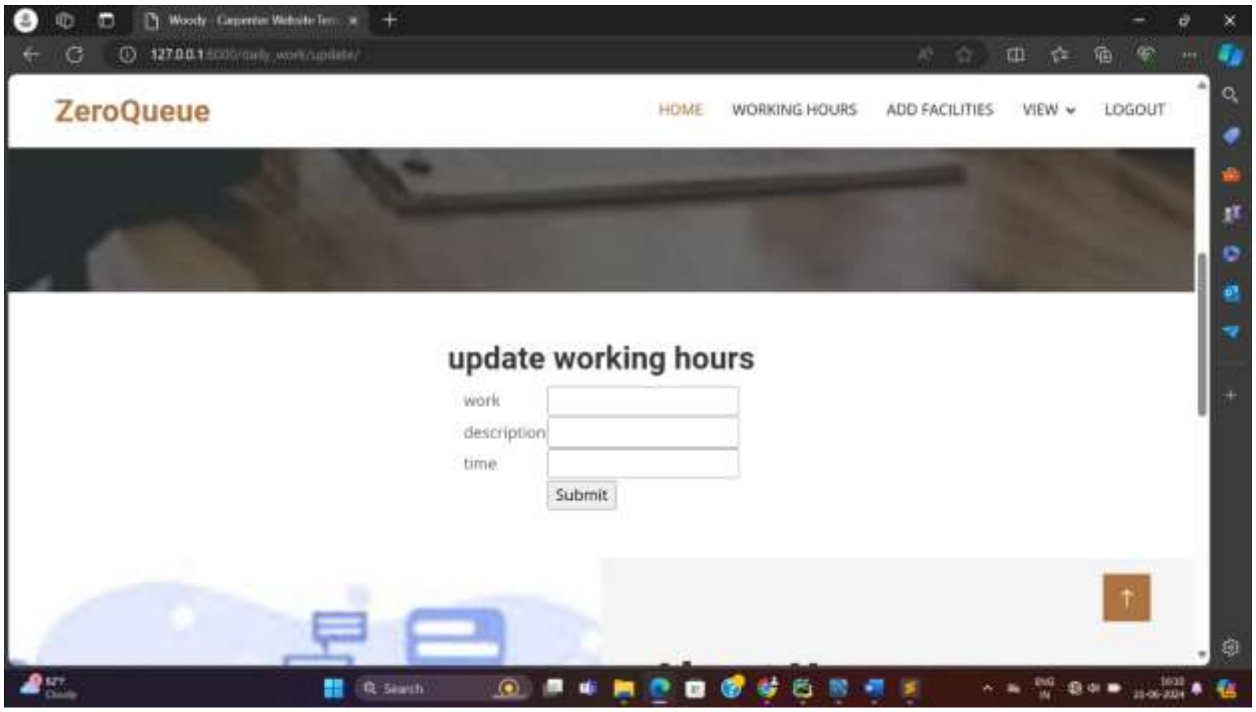
About Us

Tempor erat elitr rebum at clita. Diam dolor diam ipsum sit. Al diam amet diam et eos. Clita erat ipsum et lorem et sit, sed st lorem sit clita duo justo erat amet

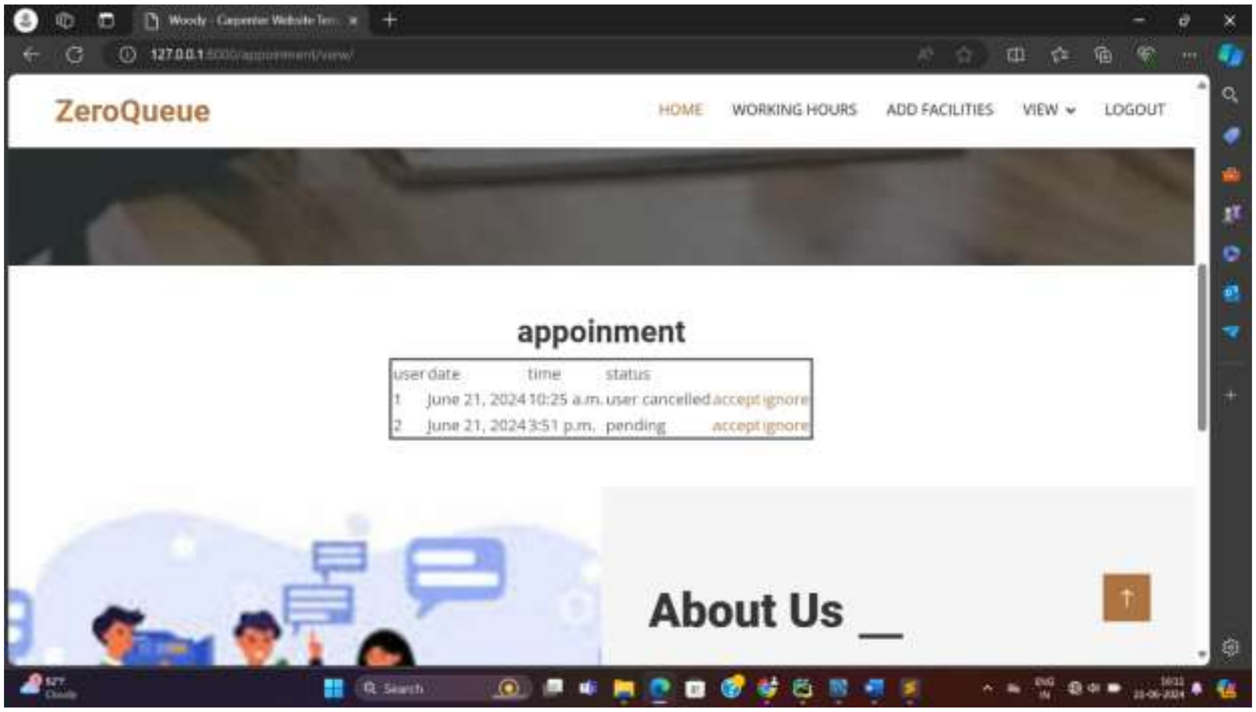
ORGANIZATION



UPDATE WORKING HOURS



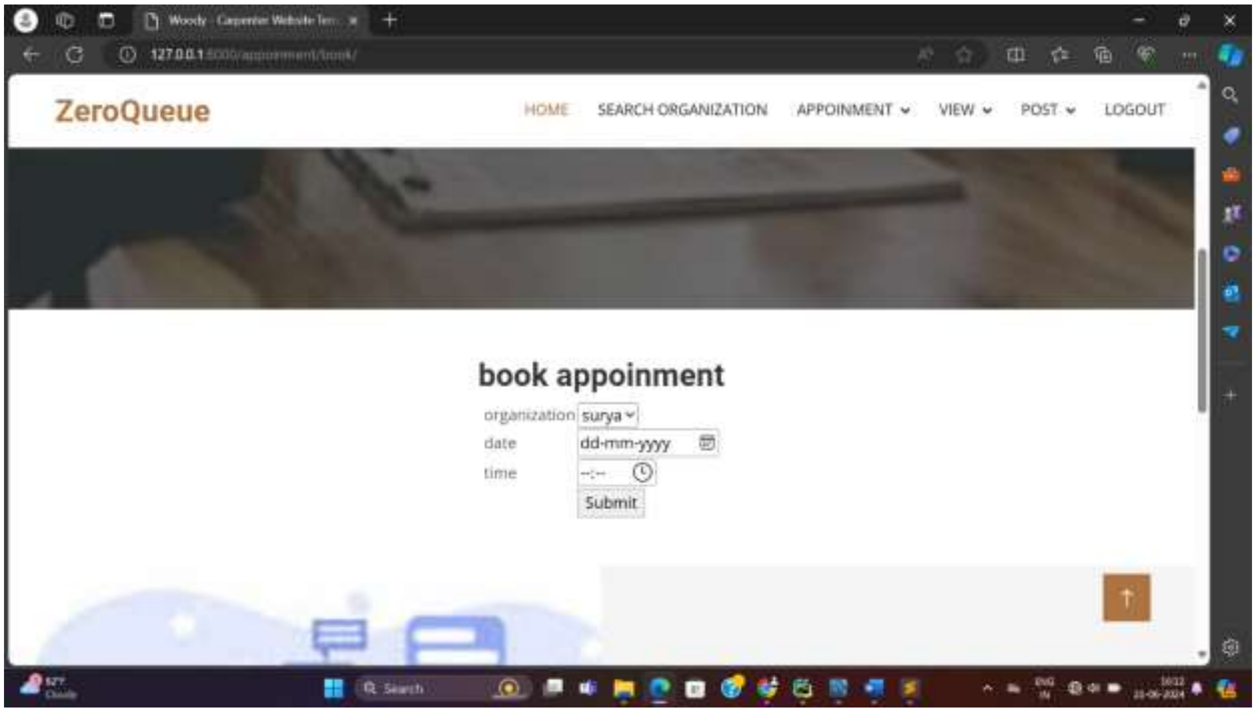
VIEW APPOINMENT



appoinment

user	date	time	status		
1	June 21, 2024	10:25 a.m.	user cancelled	accept	ignore
2	June 21, 2024	3:51 p.m.	pending	accept	ignore

BOOK APPOINMENT



book appoinment

organization	surya
date	dd-mm-yyyy
time	--:--

FUTURE SCOPE AND FURTHUR ENHANCEMENT

First, most estimates point to the relentless rise in number of EVs in the country. While some segments such as 2-wheelers and 3-wheelers would see higher sales than say, accumulatorbuses, the growth would be seen across segments including 2-Ws, 3-Ws and 4-Ws and passenger cars. Moreover, the total cost of ownership would most probably come down to equivalent levels of petrol cars as indicated by the pronouncements of the Road Transport and Highways Minister. As a result, riding on this increasing momentum in the EV market, the future of charging terminuss also looks promising.

Even as EVs increasingly go mainstream as part of the wider mobility network and usage and see enhanced uptake, the charging terminus ecosystem has been somewhat slow to develop. However, in the last one year alone, the number of charging terminuss in the country has grown more than five times. This testifies to the promise that this sector holds for ever-rising number of EV users as well as the cause of clean mobility in the country.

BIBLIOGRAPHY

BOOKS

- Rohit Khuranna , Software Engineering Principles and practices
- Elmasri and Navathe, Fundamentals of Database Systems, Addison Wesley

WEBSITES

- www.stackoverflow.com/
- http://www.mysql.com/
- http://www.codeproject.com/
- http://www.tutorialspoint.com/
- http://www.w3schools.com/

APPENDIX

According to a joint report by Indian Venture and Alternate Capital Association (IVCA), Induslaw and EY, the number of charging terminuss is expected to increase to 100,000 units by 2027 to accommodate the increasing demand by nearly 1.4 million EVs expected to be on the roads by then. Then according to another research report, India would need a humongous 20 lakh charging terminuss by 2030 to cater to a mammoth 5 crore EVs by that year.

GLOSSARY

- Howe, A. von Mayrhauser, and Mraz, R. T. Test case generation as an AI planning problem. Automated Software Engineering, 4:77-106, 1997.
- Koehler, J., Nebel, B., Hoffman, J., and Dimopoulos, Y. Extending planning graphs to an ADL subset. Lecture Notes in Computer Science, 1348:273, 1997.
- Treutner, M. F., and Ostermann, H. Evolution of Standard Web Shop Software Systems: A Review and Analysis of Literature and Market Surveys.
- Python Programming: Using Problem Solving Approach Paperback
- Software Engineering, B.E/B.Tech IV-Semester (R-17) (Anna University) Computer Science and Engineering (CSE) & IT, Latest 2020
- King, Stephen F. en Juhn-ShiuanLiou (2004), A framework for internet channel evaluation. International Journal of Information & Management 24:473–488. 1:65–75.