E-Library

Submitted by

Name of the Students:

1. SUMAN MONDAL (REG. NO.= **D181913381**)
2. DEBASMITA SHEE (REG. NO.= **D181913349**)
3. SOVAN DAS (REG. NO.= **D181913350**)
4. RIJU DAS (REG. NO.= **D181913367**)
5. SAGAR DEBNATH (REG. NO.= **D181913382**)

In partial fulfilment for the award of the degree of

DIPLOMA IN COMPUTER SCIENCE

&

TECHNOLOGY

BENGAL INSTITUTE OF TECHNOLOGY

Session :-2020-21

TABLE OF CONTENTS

S.NO. Name of the Topic

1. INTRODUCTION

A library is a place where not only books but also magazines, journals, and newspapers are well-stocked for the benefit of the readers. Besides this one can also get the entire charts, Encyclopedia, government gazette, etc. A reader can either read in the library or borrow the book/journal of his choice and take it home. A library is a popular place in the academic curriculum. With the growing popularity of the internet, the retrieval of information becomes faster. In view of the above facts, it is apparent that a library is a very important place in society

2. OBJECTIVE & SCOPE

Library Management System is a term for computer-based system that manage the catalogue of a library. The main purpose of this system is to manage library daily operation efficiently.

Objectives of Library Management System (LMS)

a) To build a system that can receive input and generate automatically output in easy way and short time.

b) To build a monitoring system that is able to monitor and manage all library operations efficiently.

c) Give an opportunity to librarians to reduce mistakes that always happen during manual method.

d) To store properly the library items in order to maintain their security.

e) To enter and preserve details of the various issues and keep a track on their returns.

The term "Digital Library" has a variety of potential meanings, ranging from a digitized collection of material that one might find in a traditional library through to the collection of all digital information along with the services that make that information useful to all possible users. As the WG discussed possible scenarios and challenge problems to drive our discussion of metrics, we found the need to come to at least a loose agreement on the scope of the digital library. This document is intended to serve that purpose.

SYSTEM ANALYSIS

3a. IDENTIFICATION OF NEED

System analysis is a process of gathering and interpreting facts, diagnosing problems and the information to recommend improvements on the system. It is a problem solving activity that requires intensive communication between the system users and system developers. System analysis or study is an important phase of any system development process .The system studies the minutest detail and gets analyzed. The system analysist plays the role of the interrogator and dwells deep into the working of the present system. The System is viewed as a whole and the input to the system are identified. The outputs from the organization are traced to the various processes. System analysis is concerned with becoming aware of the problem ,identifying the relevant and Decisional variables ,analysis and synthesizing the various factors and determining an optimal or at least a satisfactory solution or program of action.

A detailed study of the process must be made by various techniques like interviews, questionnaires etc. The data collected by these sources must be 9scrutinized to arrive to a conclusion. The conclusion is an understanding of how the system functions. This system is called the existing system .Now the existing system is subjected to close study and problem area are identified .The designer now function as a problem solver and tries to sort out the difficulties that the enterprise faces. The solution are given as

proposals .The proposal is then weighed with the existing system analytically and the best one is selected .The proposal is presented to the user for an endorsement by the user .The proposal is reviewed on user request and suitable changes are made. This is loop that ends as soon as the user is satisfied with proposal.

3b. FEASIBILITY STUDY

Feasibility study is made to see if the project on completion will serve the purpose the organization for the amount of work.

Effort and the time that spend on it. Feasibility study lets the developer foresee the future of the project and the usefulness. A Feasibility study of a system proposal is according to its workability, which is the impact on the organization, ability to meet their user needs and effective use of resources .Thus when a new application is proposed it normally goes through a feasibility study before it is approved for development.

The document provide the feasibility of the project that is being designed and lists various area that were considered very carefully during the feasibility study of this project such as Technical , Economic and operational feasibilities.

3c. WORK FLOW

This Document plays a vital role in the development life cycle (SDLC) as it describes the complete requirement of the system. It is meant for use by the developers and will be the basic during testing phase. Any changes made to the requirements in the future will have to go through formal change approval process.

The Waterfall Model was first Process Model to be introduced. It is also referred to as a linear-sequential life cycle model. It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases.

Waterfall model is the earliest SDLC approach that was used for software development .

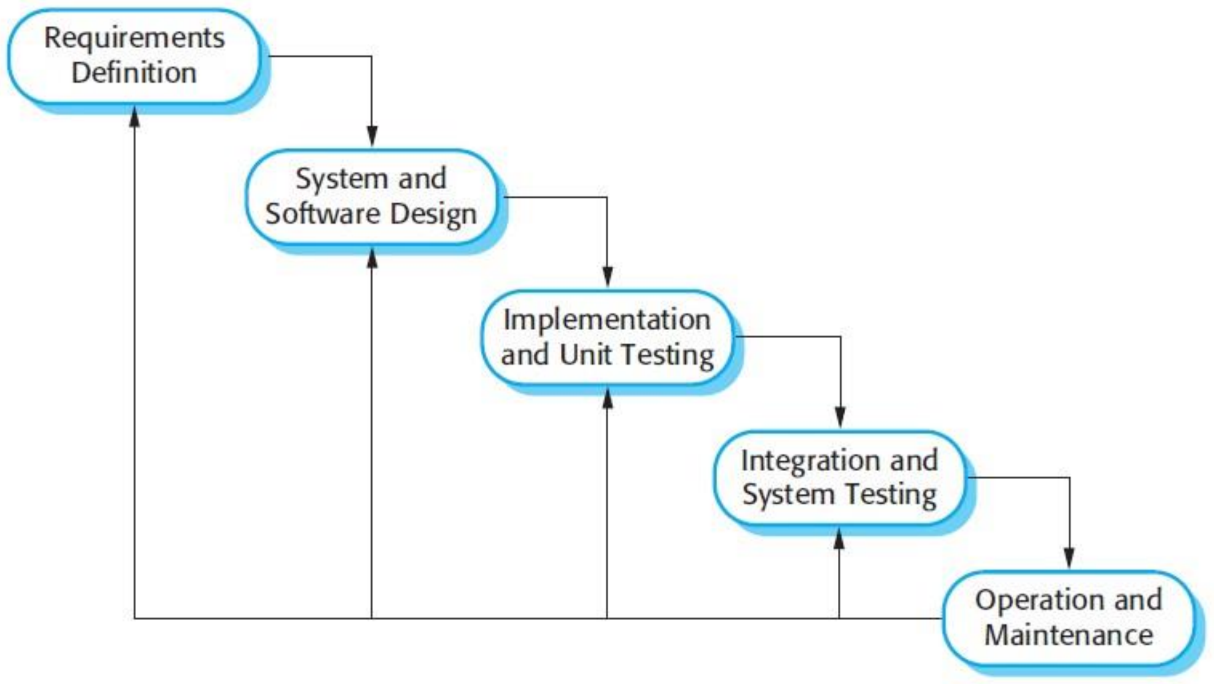
The waterfall Model illustrates the software development process in a linear sequential flow; hence it is also referred to as a linear-sequential life cycle model. This means that any phase in the development process begins only if the previous phase is complete. In waterfall model phases do not overlap.

Waterfall Model design

Waterfall approach was first SDLC Model to be used widely in Software Engineering to ensure success of the project. In "The Waterfall" approach, the whole

process of software development is divided into separate phases. In Waterfall model, typically, the

outcome of one phase acts as the input for the next phase sequentially.
Following is a diagrammatic representation of different phases of waterfall model.


Requirements Definition
System and Software Design
Implementation and Unit Testing
Integration and System Testing
Operation and Maintenance

The sequential phases in Waterfall model are:

- **Requirement Gathering and analysis:** All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification doc.
- **System Design:** The requirement specifications from first phase are studied in this phase and system design is prepared. System Design helps in specifying hardware and system requirements and also helps in defining overall system architecture.
- **Implementation:** With inputs from system design, the system is first developed in small programs called units, which are integrated in the

next phase. Each unit is developed and tested for its functionality which is referred to as Unit Testing.

- **Integration and Testing:** All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
- **Deployment of system:** Once the functional and non functional testing is done, the product is deployed in the customer environment or released into the market.
- **Maintenance:** There are some issues which come up in the client environment. To fix those issues patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

All these phases are cascaded to each other in which progress is seen as flowing steadily downwards (like a waterfall) through the phases. The next phase is started only after the defined set of goals are achieved for previous phase and it is signed off, so the name "Waterfall Model". In this model phases do not overlap.

Waterfall Model Application

Every software developed is different and requires a suitable SDLC approach to be followed based on the internal and external factors. Some situations where the use of Waterfall model is most appropriate are:

- Requirements are very well documented, clear and fixed.
- Product definition is stable.

- Technology is understood and is not dynamic.
- There are no ambiguous requirements.

- Ample resources with required expertise are available to support the product.
- The project is short.

The advantage of waterfall development is that it allows for departmentalization and control. A schedule can be set with deadlines for each stage of development and a product can proceed through the development process model phases one by one.

Development moves from concept, through design, implementation, testing, installation, troubleshooting, and ends up at operation and maintenance. Each phase of development proceeds in strict order.

3d. STUDY OF THE SYSTEM

Modules:

The modules used in this software are as follows:

- LOGIN: This module is for LIBRARIAN,ADMIN ADMIN has the authority to Insert and view issue books and library data details in the other side the librarian has the authority to issue book and view book for the members, who wants to buy any of the book available in the library.

- HOME :This page contains an overview of highlights for other pages.
- ABOUT-US:This page contains the information about E-LIBRARY administrator.

- CONTACT US :This page contains the information about us .Any member, who are using this library can send any feedback or suggestion for add books and complain .

- OUR BOOKS: This page contains the updated information about the stock of our library books.

3e. INPUT AND OUTPUT

- The main inputs ,outputs and the major function the details are

INPUT

- Librarian can login using librarian-id and password.

Admin can login using admin-id and password.

- Admin insert and modify the details librarian,our books,catagory, and library data details.
- Councilor can take order and view books,as a result issue books.
- Councillor can member student details searching them by library id-card.

OUTPUT

- Members can view the details of books, price, library card price, issue date-due date.
- Admin can view the details of librarian,our books,catagory, and library data details.

3f. SOFTWARE REQUIREMENT SPECIFICATIONS

Software Requirements Specification provides an overview of the entire project. It is a description of a software system to be developed, laying out functional and nonfunctional requirements. 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 project to be developed. This is prepared after the detailed communication with project team and the customer.

The developer is responsible for:-

- ✓ Developing the system, which meets the SRS and solving all the requirements of the system?
- ✓ Demonstrating the system and installing the system at client’s location after acceptance testing is successful.
- ✓ Submitting the required user manual describing the system interfaces to work on it and also the documents of the system.
- ✓ Conducting any user training that might be needed for using the system.
- ✓ Maintain the system for a period of one year after installation.

HARDWARE REQUIREMENTS:

- Laptop or Computer that has a 1.6GHz or faster processor

- 4 GB (64 Bit) RAM

- HDD 50 GB Hard Disk Space and Above Hardware Requirements
- 7200 RPM hard disk drive
- PEN Drive

SOFTWARE REQUIREMENTS:

•WINDOWS OS (VER-10)

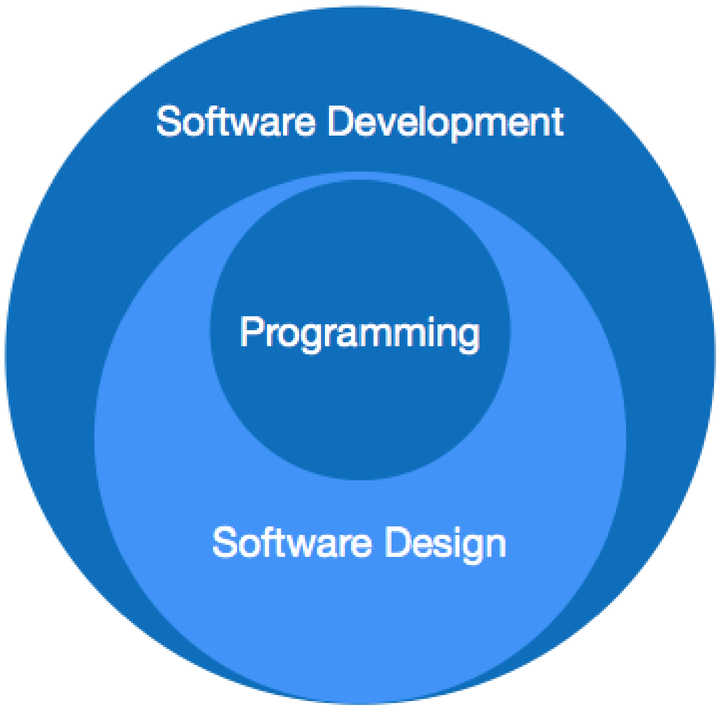
•MySql Server

•Xampp server

•VS Code Editor

3g. SOFTWARE ENGINEERING PARADIGM APPLIED

Software paradigms refer to the methods and steps, which are taken while designing the software. There are many methods proposed and are in work today, but we need to see where in the software engineering these paradigms stand. These can be combined into various categories, though each of them is contained in one another.



Programming paradigm is a subset of Software design paradigm which is further a subset of Software development paradigm.

There are two levels of reliability. The first is meeting the right requirement. A carefully and through systems

study is needed to satisfy this aspect of reliability. The second level of systems reliability involves the actual working delivered to the user. At this level, the systems reliability is interwoven with software engineering and development. There are three approaches to reliability.

1. Error avoidance: Prevents errors from occurring in software.

2. Error detection and correction: In this approach errors are recognized whenever they are encountered and correcting the error by effect of error of the system does not fail.

3. Error tolerance: In this approach errors are recognized whenever they occur, but enables the system to keep running through degraded perform
or Appling values that instruct the system to continue process.

<u>Maintenance:</u>

The key to reducing need for maintenance, while working, if possible to do essential tasks.

1. More accurately defining user requirement during system development.

2. Assembling better systems documents.

3. Using some effective methods for designing, processing, and login and communicating information with project team members.

4. Making better use of existing tools and techniques.

SYSTEM DESIGN

4a. DATA FLOW DIAGRAM

A **data flow diagram** (**DFD**) is a graphical representation of the "flow" of data through an information system, modelling its process aspects. A **DFD** is often used as a preliminary step to create an overview of the system, which can later be elaborated.

DFDs can also be used for the visualization of data processing (structured design).

A DFD shows what kind of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of process or information about whether processes will operate in sequence or in parallel (which is shown on a flowchart).

This context-level DFD is next "exploded", to produce a Level 1 DFD that shows some of the detail of the system being modeled. The Level 1 DFD shows how the system is divided into sub-systems (processes), each of whi1ch deals with one or more of the data flows to or from an external agent, and which together provide all of the functionality of the system as a whole. It also identifies internal data stores that must be present in order for the system to do its job, and shows the flow of data between the various parts of the system.

Data flow diagrams are one of the three essential perspectives of the structured-systems analysis and design method SSADM. The sponsor of a project and the end users will need to be briefed and consulted throughout all stages of a system's evolution. With a

data flow diagram, users are able to visualize how the system will operate, what the system will accomplish, and how the system will be implemented. The old system's dataflow diagrams can be drawn up and compared with the new system's data flow diagrams to draw comparisons to implement a more efficient system. Data flow diagrams can be used to provide the end user with a physical idea of where the data they input ultimately has an effect upon the structure of the whole system from order to dispatch to report.

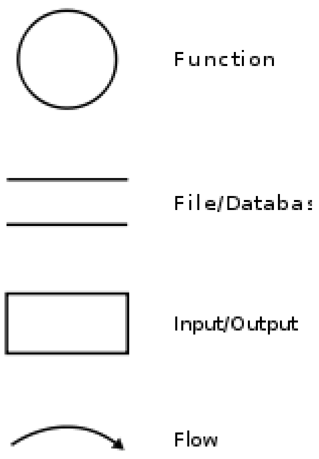
How any system is developed can be determined through a data flow diagram model.

In the course of developing a set of *leveled* data flow diagrams the analyst/designer is forced to address how the system may be decomposed into component sub-systems, and to identify the transaction data in the data model.

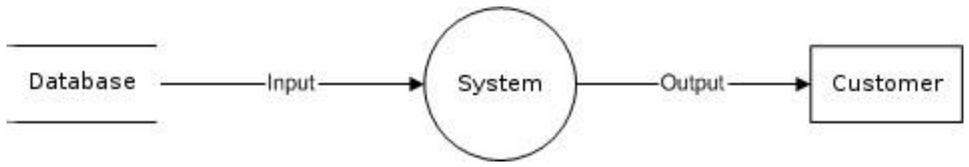
Data flow diagrams can be used in both Analysis and Design phase of the SDLC.

There are different notations to draw data flow diagrams. defining different visual representations for processes, data stores, data flow, and external entities.[6]

DFD NOTATION



DFD EXAMPLE



<u>Steps to Construct Data Flow Diagram:-</u>

Four Steps are generally used to construct a DFD.

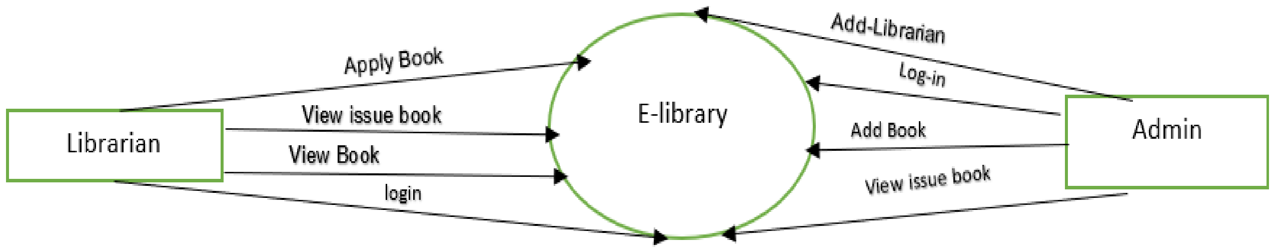
- Process should be named and referred for easy reference. Each name should be representative of the reference.
- The destination of flow is from top to bottom and from left to right.
- When a process is distributed into lower level details they are numbered.
- The names of data stores, sources and destinations are written in capital letters.

Rules for constructing a Data Flow Diagram:-

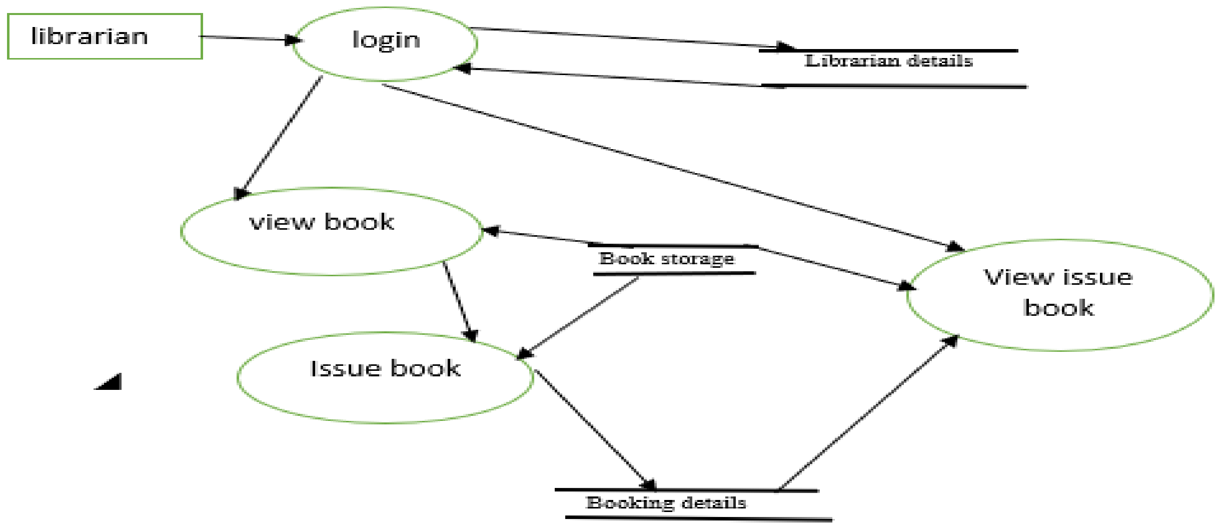
- Arrows should not cross each other.
- Squares, Circles, Files must bear a name.
- Decomposed data flow squares and circles can have same names. □ Draw all data flow around the outside of the diagram.

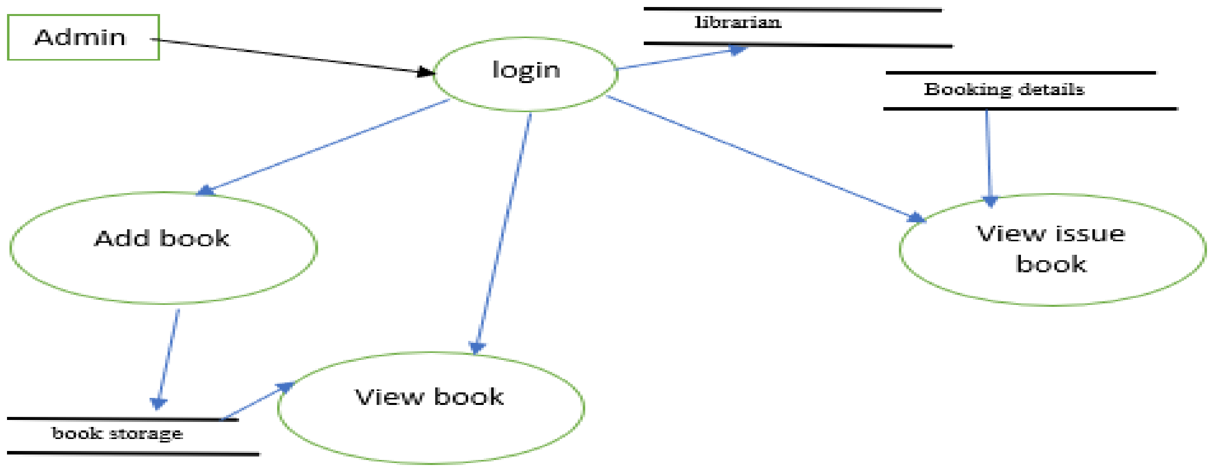
DATA FLOW DIAGRAM

LEVEL-0 DFD DIAGRAM



LEVEL -1 DFD DIAGRAM



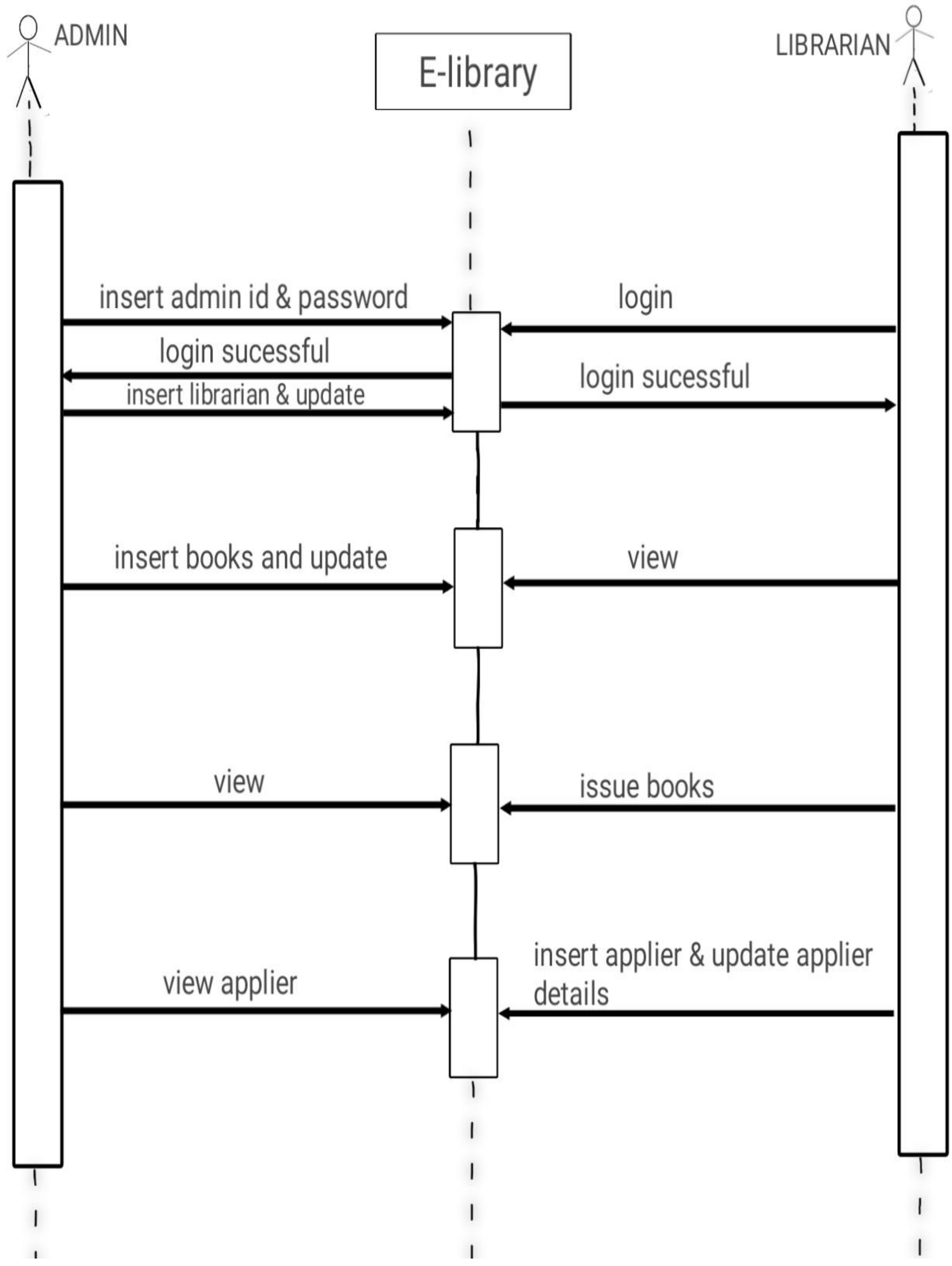


4b. SEQUENCE DIAGRAM

A **Sequence diagram** is an interaction diagram that shows how processes operate with one another and what is their order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called **event diagrams** or **event scenarios**.

A sequence diagram shows, as parallel vertical lines (*lifelines*), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

Sequence diagram is the most common kind of **interaction diagram**, which focuses on the **message** interchange between a number of **lifelines**.
Sequence diagram describes an interaction by focusing on the sequence of messages that are exchanged, along with their corresponding occurrence specifications on the lifelines.
The following nodes and edges are typically drawn in a **UML sequence diagram**: **lifeline**, **execution-specification**, **message**, fragment, interaction, **state invariant**, **continuation**, **destruction occurrence**.


ADMIN
E-library
LIBRARIAN
insert admin id & password
login
login sucessful
login sucessful
insert librarian & update
insert books and update
view
view
issue books
view applier
insert applier & update applier details

4c. ENTITY RELATIONSHIP DIAGRAM

In software engineering, an **entity–relationship model**(**ER model**) is a data model for describing the data or information aspects of a business domain or its process requirements, in an abstract way that lends itself to ultimately being implemented in a database such as a relational. The main components of ER models are entities (things) and the relationships that can exist among them.

An entity–relationship model is the result of using a systematic process to describe and define a subject area of business data. It does not define business process; only visualize business data. The data is represented as components (entities) that are linked with each other by relationships that express the dependencies and requirements between them, such as: one building may be divided into zero or more apartments, but one apartment can only be located in one building. Entities may have various properties (attributes) that characterize them. Diagrams created to represent these entities, attributes, and relationships graphically are called entity–relationship diagrams.

An ER model is typically implemented as a database. In the case of a relational database, which stores data in tables, every row of each table represents one instance of an entity. Some data fields in these tables point to indexes in other tables; such pointers are the physical implementation of the relationships.

The three schema approach to software engineering uses three levels of ER models that may be developed.

Conceptual data model

The conceptual ER model normally defines master reference data entities that are commonly used by the organization. Developing an enterprise-wide conceptual ER model is useful to support documenting the data architecture for an organization.A conceptual ER model may be used as the foundation for one or more logical data models . The purpose of the conceptual ER model is then to establish structural metadata commonality for the master data entities between the set of logical ER models. The conceptual data model may be used to form commonality relationships between ER models as a basis for data model integration.

Logical data model

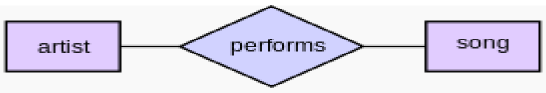
The logical ER model contains more detail than the conceptual ER model. In addition to master data entities, operational and transactional data entities are now defined. The details of each data entity are developed and the relationships between these data entities are established. The logical ER model is however developed independent of technology into which it can be implemented.

Physical data model

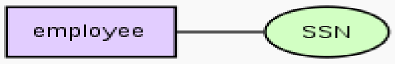
One or more physical ER models may be developed from each logical ER model. The physical ER model is normally developed to be instantiated as a database. Therefore, each physical ER model must contain enough detail to produce a database and each physical ER model is technology dependent since each database management system is somewhat different. The physical model is normally instantiated in the structural metadata of a database management system as relational database objects such as database tables, database indexes such as unique keyindexes, and database constraints such as a foreign key constraint or a commonality constraint. The ER model is also normally used to design modifications to the relational database objects and to maintain the structural metadata of the database.

The first stage of information system design uses these models during the requirements analysis to describe information needs or the type of information that is to be stored in a database. The data modelingtechnique can be used to describe any ontology (i.e. an overview and classifications of used terms and their relationships) for a certain area of interest. In the case of the design of an information system that is based on a database, the conceptual data model is, at a later stage (usually called logical design), mapped to a logical data model, such as the relational model; this in turn is mapped to a physical model during physical design. Note that sometimes, both of these phases are referred to as "physical design". It is also used in database management system.

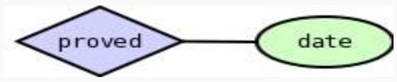
Entity–relationship modeling



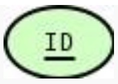
Two related entities



An entity with an attribute



A relationship with an attribute

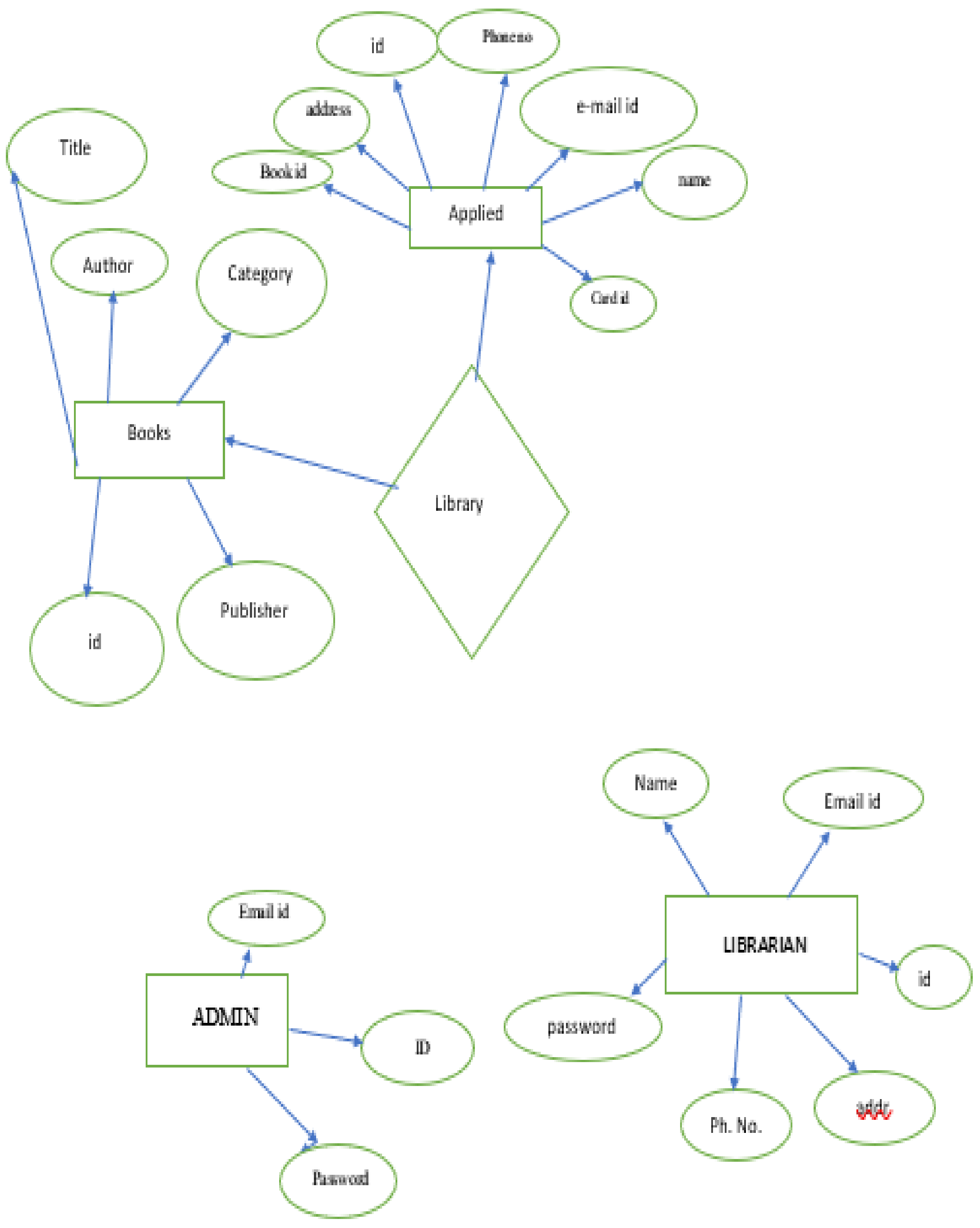


Primary key

Cardinality constraints are expressed as follows:

- A double line indicates a *participation constraint*, totality or subjectivity : all entities in the entity set must participate in *at least one* relationship in the relationship set;
- an arrow from entity set to relationship set indicates a key constraint, i.e. injectivity: each entity of the entity set can participate in *at most one* relationship in the relationship set;
- A thick line indicates both, i.e. bijectivity: each entity in the entity set is involved in *exactly one* relationship.
- An underlined name of an attribute indicates that it is a key: two different entities or relationships with this attribute always have different values for this attribute.

ER DIAGRAM



id
Phoneno
address
e-mail id
Title
Book id
name
Applied
Author
Category
Card id
Books
Library
id
Publisher
Name
Email id
Email id
LIBRARIAN
id
ADMIN
password
ID
Ph. No.
addr.
Pasword

4d. USE CASE DIAGRAM

A **use case diagram** at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well.

So only static behavior is not sufficient to model a system rather dynamic behavior is more important than static behavior. In UML there are five diagrams available to model 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. So use case diagrams are 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.

So to model the entire system numbers of use case diagrams are used.

The purpose of use case diagram is to capture the dynamic aspect of a system. But this definition is too generic to describe the purpose.

Because other four diagrams (activity, sequence, collaboration and State chart) are also having the same purpose. So we will look into some specific purpose which will distinguish it from other four diagrams.

Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. So when a system is analyzed to gather its functionalities use cases are prepared and actors are identified.

Now when the initial task is complete use case diagrams are modelled to present the outside view.

So in brief, the purposes of use case diagrams can be as follows:

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

How to draw Use Case Diagram?

Use case diagrams are considered for high level requirement analysis of a system. So when the requirements of a system are analyzed the functionalities are captured in use cases.

So we can say that uses cases are nothing but the system functionalities written in an organized manner. Now the second things which are relevant to the use cases are the actors. Actors can be defined as something that interacts with the system.

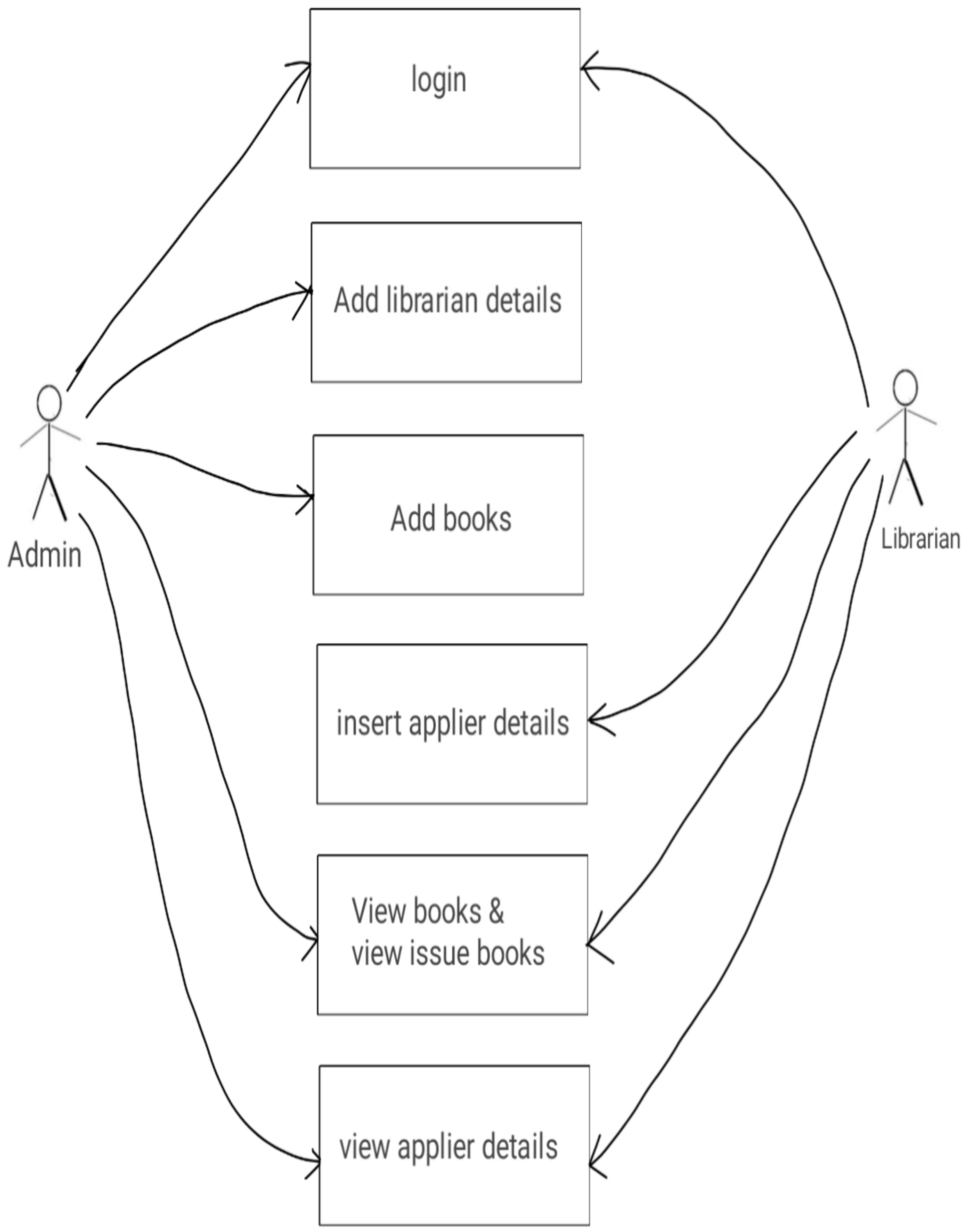
The actors can be human user, some internal applications or may be some external applications. So in a brief when we are planning to draw a use case diagram we should have the following items identified.

- Functionalities to be represented as an use case
- Actors
- Relationships among the use cases and actors.

Use case diagrams are drawn to capture the functional requirements of a system. So after identifying the above items we have to follow the following guidelines to draw an efficient use case diagram.

- The name of a use case is very important. So the name should be chosen in such a way so that it can identify the functionalities performed.
- Give a suitable name for actors.
- Show relationships and dependencies clearly in the diagram.
- Do not try to include all types of relationships. Because the main purpose of the diagram is to identify requirements.
- Use note whenever required to clarify some important points.

USE CASE DIAGRAM



4.e MODULARIZATION DETAILS

As Modularization has gained increasing focus from companies outside its traditional industries of aircraft and automotive, more and more companies turn to it as strategy and product development tool. I intend to explain the importance aspects of modularization and how it should be initiated within a company. After determining the theoretical steps of modularization success described in literature, I intend to conduct a multiple case study of companies who have implemented modularization in order to find how real world modularization was initiated and used to improve the company's competitiveness. By combining theory and practical approach to modularization I will derive at convergence and divergence between theoretical implementation to modularization and real world implementation to modularization. This gives a valuable input for both implantations in companies as well as new aspects to be further.

DATA INTEGRITY AND CONSTRAINTS

Data integrity is normally enforced in a database system by a series of integrity constraints or rules. Three types of integrity constraints are an inherent part of the relational data model: entity integrity, referential integrity and domain integrity:

- *Entity integrity* concerns the concept of a primary key. Entity integrity is an integrity rule which states that every table must have a primary key and that the column or columns chosen to be the primary key should be unique and not null.

- Concerns the concept of a foreign key. The referential integrity rule states that any foreign-key value can only be in one of two states. The usual state of affairs is that the foreign-key value refers to a primary key value of some table in the database.

Occasionally, and this will depend on the rules of the data owner, a foreign-key value can be null. In this case we are explicitly saying that either there is no relationship between the objects represented in the database or that this relationship is unknown.

- *Domain integrity* specifies that all columns in a relational database must be declared upon a defined domain. The primary unit of data in the relational data model is the data item. Such data items are said to be non-decomposable or atomic. A domain is a set of values of the same type.

4f. DATABASE DESIGN

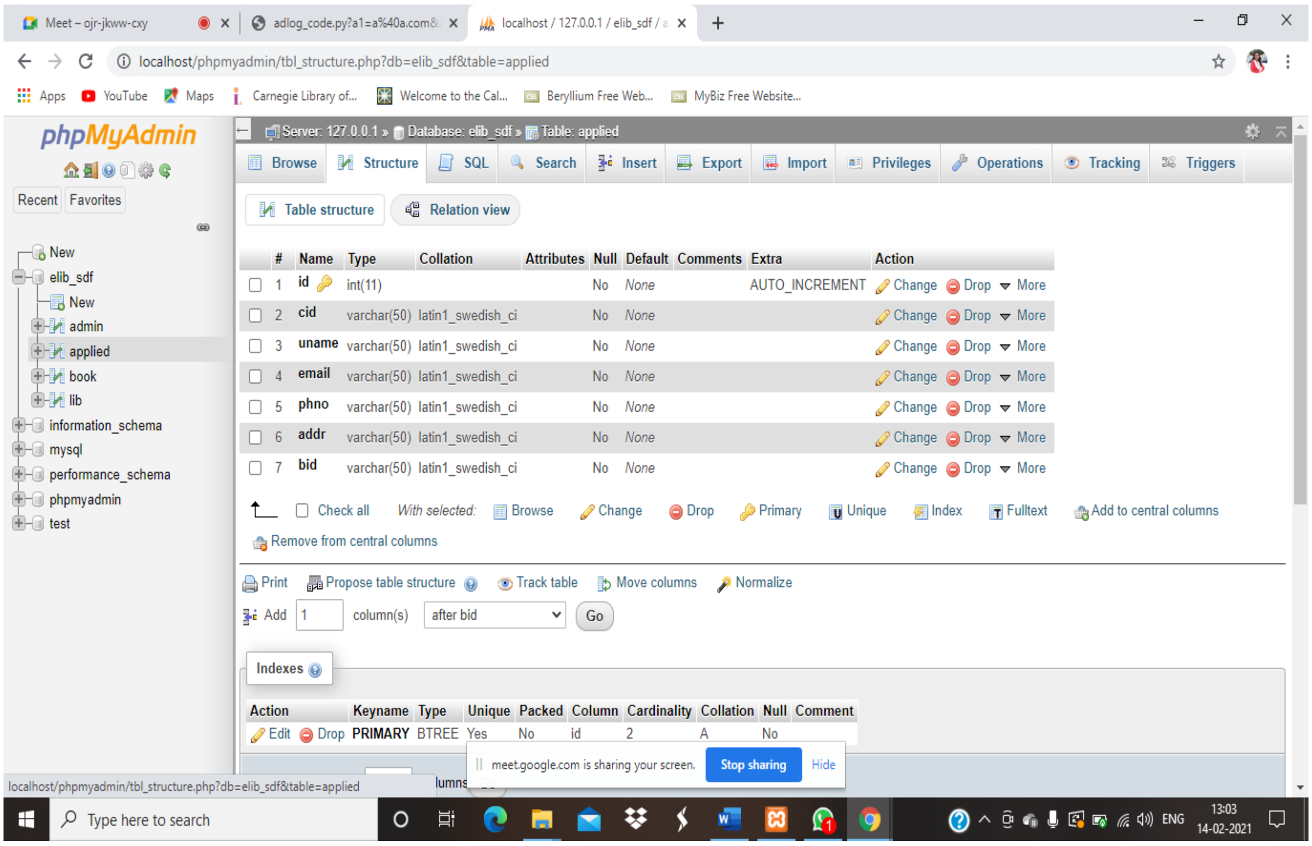
A database is an organized mechanism that has capability of storing information through which a user can retrieve stored information in an effective and efficient manner. The data is the purpose of any database and must be protected.

The database design is two level processes. In the first step, user requirements are gathered together and a database is designed which will meet these requirements as clearly as possible. This step is called information Level design and it is taken independent of any individual DBMS.

In the following snapshots we display the way we have used SQL Server as the back-end RDBMS for our project and the various entities that have been used along with their table definition and table data.

DATA DICTIONARY

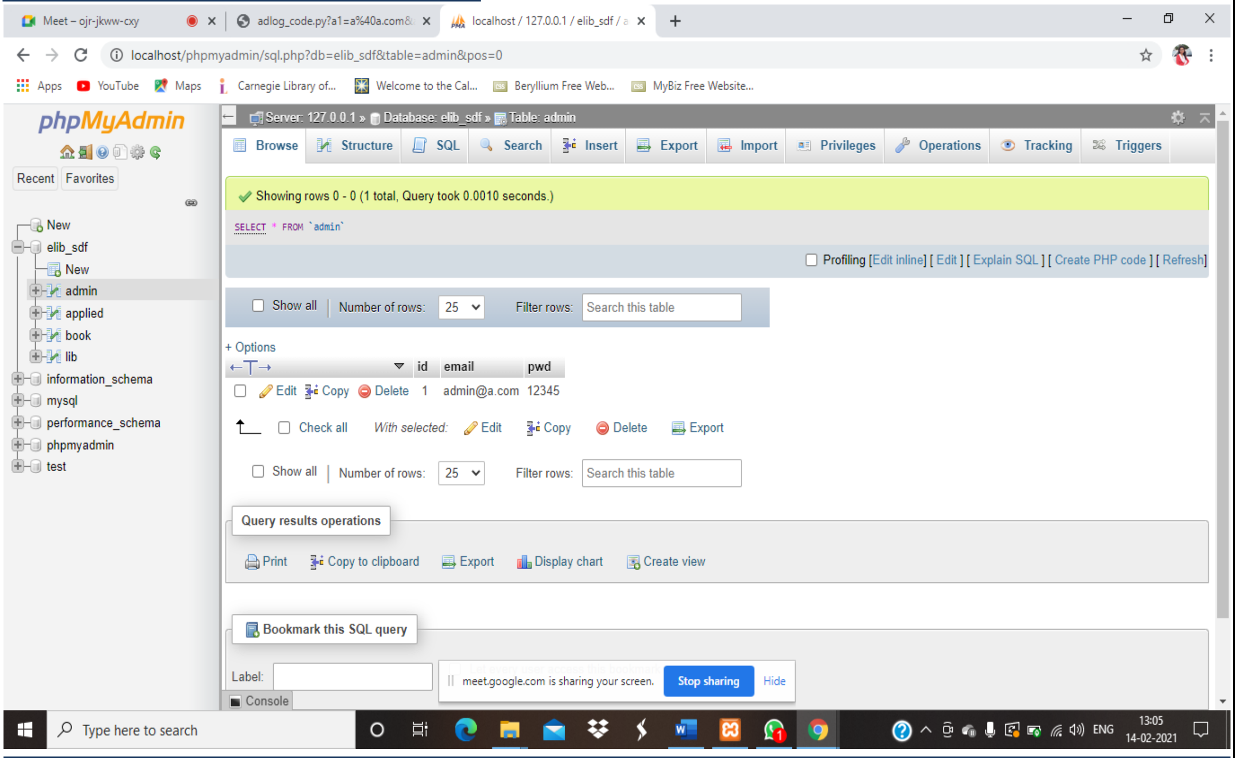
Applier Registration form table



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(11)			No	None		AUTO_INCREMENT	Change Drop More
2	cid	varchar(50)	latin1_swedish_ci		No	None			Change Drop More
3	uname	varchar(50)	latin1_swedish_ci		No	None			Change Drop More
4	email	varchar(50)	latin1_swedish_ci		No	None			Change Drop More
5	phno	varchar(50)	latin1_swedish_ci		No	None			Change Drop More
6	addr	varchar(50)	latin1_swedish_ci		No	None			Change Drop More
7	bid	varchar(50)	latin1_swedish_ci		No	None			Change Drop More

Action	Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
Edit Drop	PRIMARY	BTREE	Yes	No	id	2	A	No	

Admin login table data



localhost/phpmyadmin/sql.php?db=elib_sdf&table=admin&pos=0

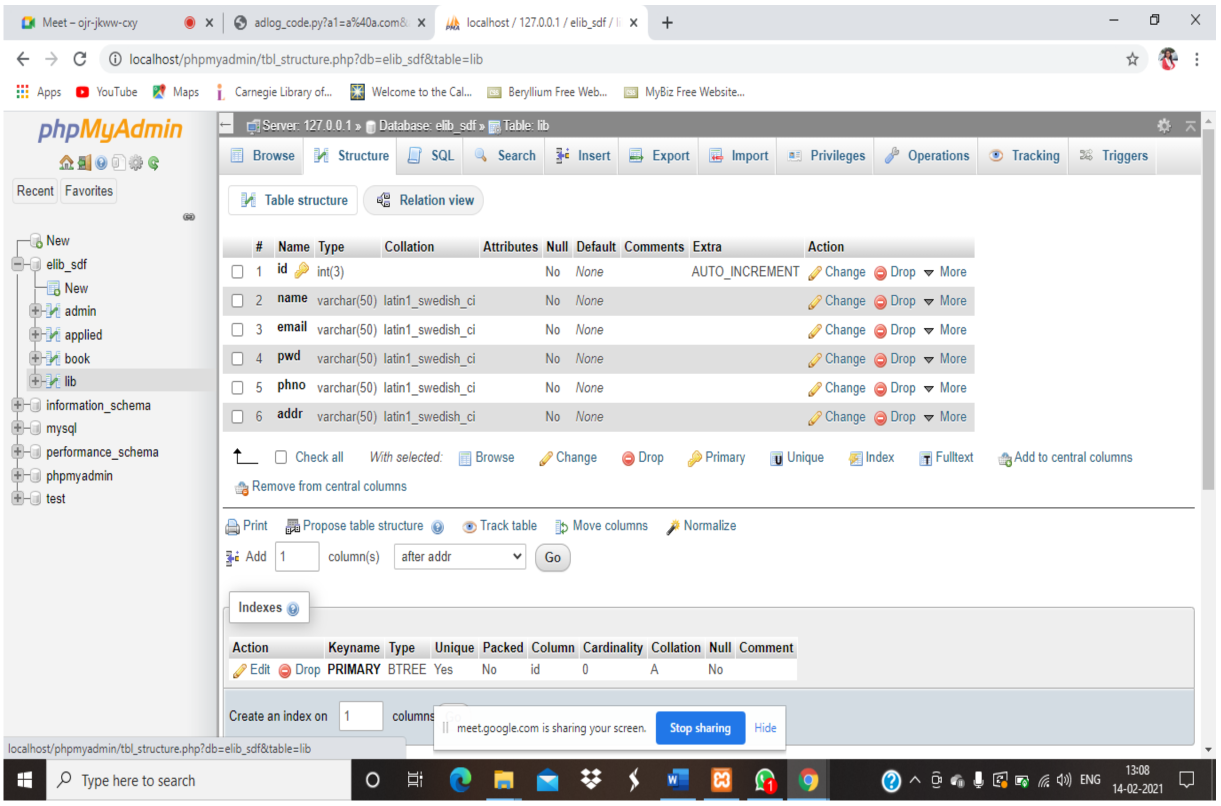
Server: 127.0.0.1 » Database: elib_sdf » Table: admin

✓ Showing rows 0 - 0 (1 total, Query took 0.0010 seconds.)

```
SELECT * FROM `admin`
```

	id	email	pwd
Edit Copy Delete	1	admin@a.com	12345

Librarian registration table



localhost/phpmyadmin/tbl_structure.php?db=elib_sdf&table=lib

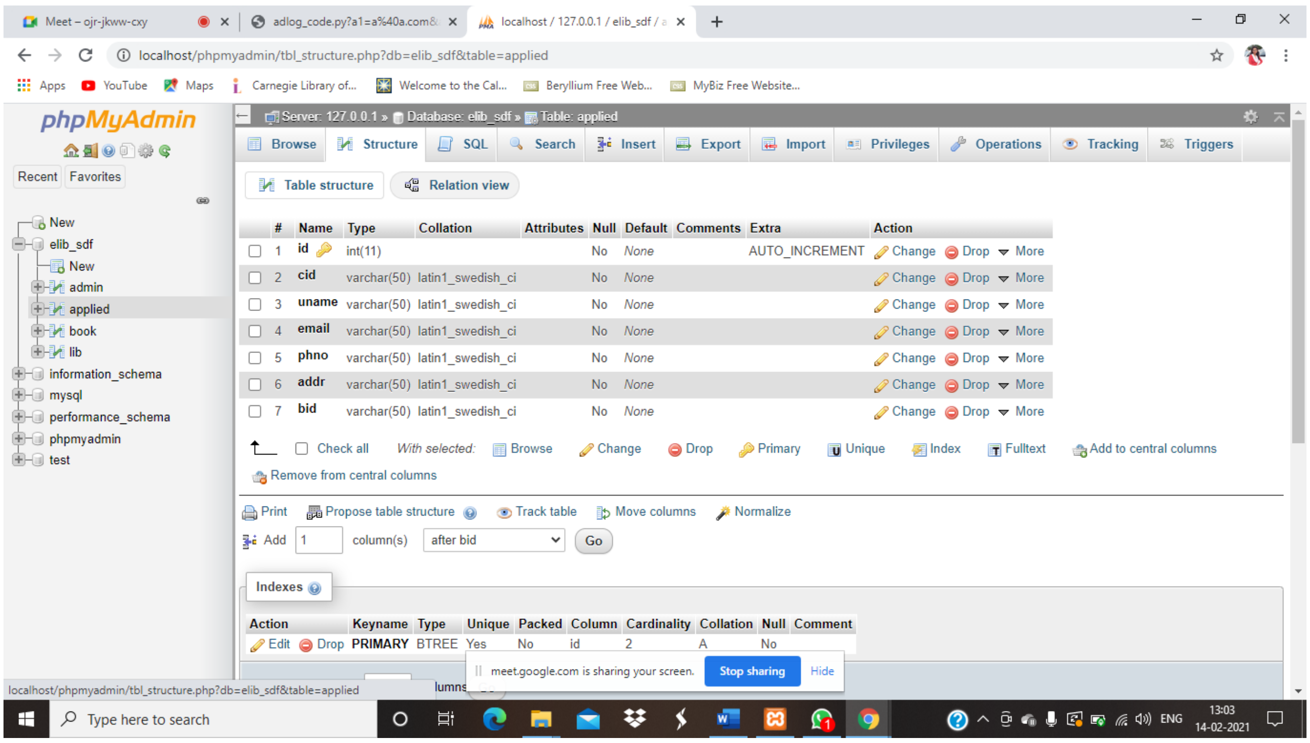
Server: 127.0.0.1 » Database: elib_sdf » Table: lib

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(3)			No	None		AUTO_INCREMENT	Change Drop More
2	name	varchar(50)	latin1_swedish_ci		No	None			Change Drop More
3	email	varchar(50)	latin1_swedish_ci		No	None			Change Drop More
4	pwd	varchar(50)	latin1_swedish_ci		No	None			Change Drop More
5	phno	varchar(50)	latin1_swedish_ci		No	None			Change Drop More
6	addr	varchar(50)	latin1_swedish_ci		No	None			Change Drop More

Indexes

Action	Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
Edit Drop	PRIMARY	BTREE	Yes	No	id	0	A	No	

Applier table

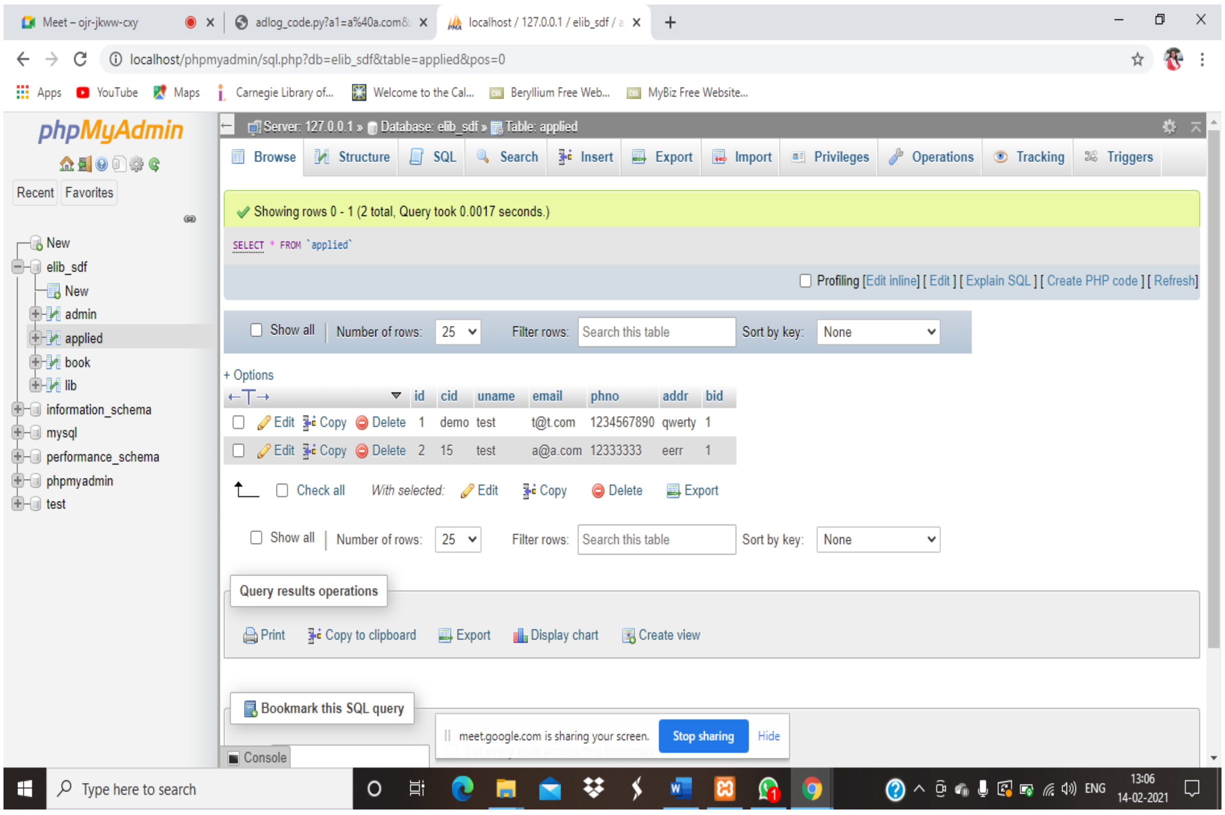


#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(11)			No	None		AUTO_INCREMENT	Change Drop More
2	cid	varchar(50)	latin1_swedish_ci		No	None			Change Drop More
3	uname	varchar(50)	latin1_swedish_ci		No	None			Change Drop More
4	email	varchar(50)	latin1_swedish_ci		No	None			Change Drop More
5	phno	varchar(50)	latin1_swedish_ci		No	None			Change Drop More
6	addr	varchar(50)	latin1_swedish_ci		No	None			Change Drop More
7	bid	varchar(50)	latin1_swedish_ci		No	None			Change Drop More

Indexes

Action	Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
Edit Drop	PRIMARY	BTREE	Yes	No	id	2	A	No	

Applier Table Data

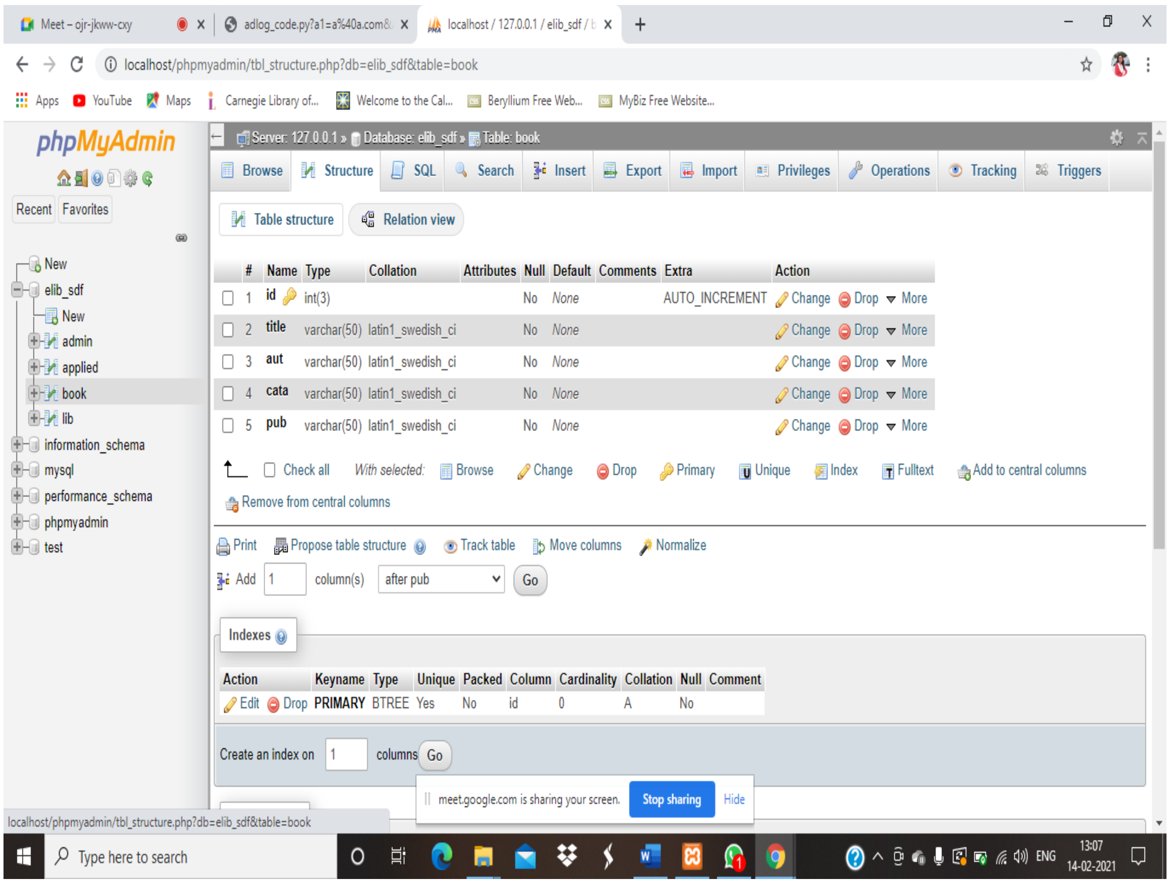

localhost/phpmyadmin/sql.php?db=elib_sdf&table=applied&pos=0
Server: 127.0.0.1 » Database: elib_sdf » Table: applied
Browse
Structure
SQL
Search
Insert
Export
Import
Privileges
Operations
Tracking
Triggers

Showing rows 0 - 1 (2 total, Query took 0.0017 seconds.)

```
SELECT * FROM `applied`
```

	id	cid	uname	email	phno	addr	bid
Edit Copy Delete	1	demo	test	t@t.com	1234567890	qwerty	1
Edit Copy Delete	2	15	test	a@a.com	12333333	eerr	1

Books Table



	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action		
☐	1	id	int(3)			No	None		AUTO_INCREMENT	Change	Drop	More
☐	2	title	varchar(50)	latin1_swedish_ci		No	None			Change	Drop	More
☐	3	aut	varchar(50)	latin1_swedish_ci		No	None			Change	Drop	More
☐	4	cata	varchar(50)	latin1_swedish_ci		No	None			Change	Drop	More
☐	5	pub	varchar(50)	latin1_swedish_ci		No	None			Change	Drop	More

Action		Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
Edit	Drop	PRIMARY	BTREE	Yes	No	id	0	A	No	

OUTPUT **SCREEN**

5a. USER INTERFACE DESIGN

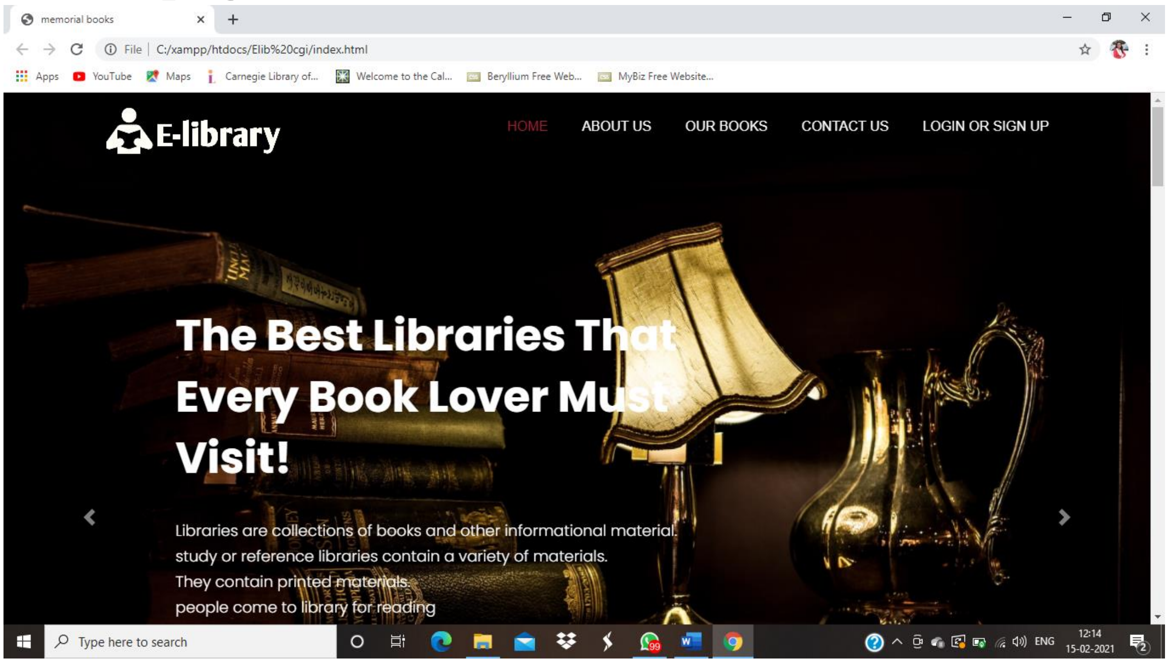
User interface design (UID) or **user interface engineering** is the design of user interfaces for machines and software, such as computers, home appliances, mobile devices, and other electronic devices, with the focus on maximizing the user experience. The goal of user interface design is to make the user's interaction as simple and efficient as possible, in terms of accomplishing user goals (usercentered design).

Good user interface design facilitates finishing the task at hand without drawing unnecessary attention to it. Graphic design and typography are utilized to support its usability, influencing how the user performs certain interactions and improving the aesthetic appeal of the design; design aesthetics may enhance or detract from the ability of users to use the functions of the interface. The design process must balance technical functionality and visual elements (e.g., mental model) to create a system that is not only operational but also usable and adaptable to changing user needs.

Interface design is involved in a wide range of projects from computer systems, to cars, to commercial planes; all of these projects involve much of the same basic human interactions yet also require some unique skills and knowledge. As a result, designers tend to specialize in certain types of projects and have skills centered on their expertise, whether that be software design, user research, web design, or industrial design.

SNAPSHOTS

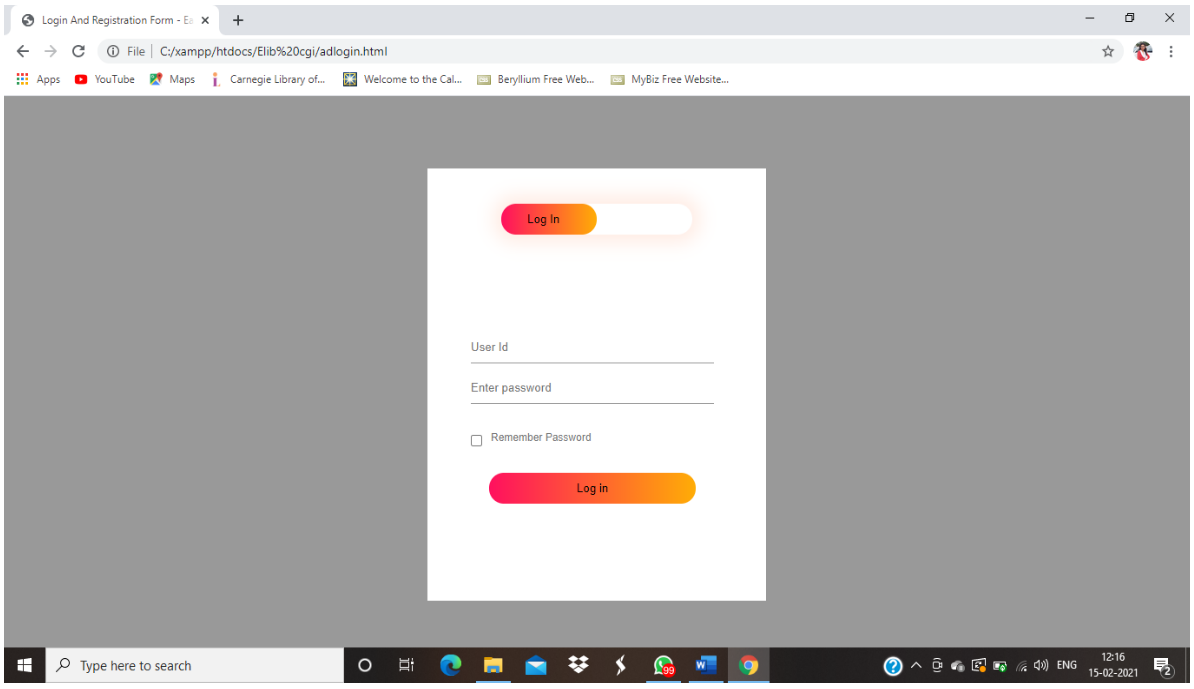
Home page



The Best Libraries That Every Book Lover Must Visit!

Libraries are collections of books and other informational material.
study or reference libraries contain a variety of materials.
They contain printed materials.
people come to library for reading

Admin login page


Login And Registration Form
C:/xampp/htdocs/Elib%20cgi/adlogin.html
Log In
User Id
Enter password
Remember Password
Log in

Admin_login code

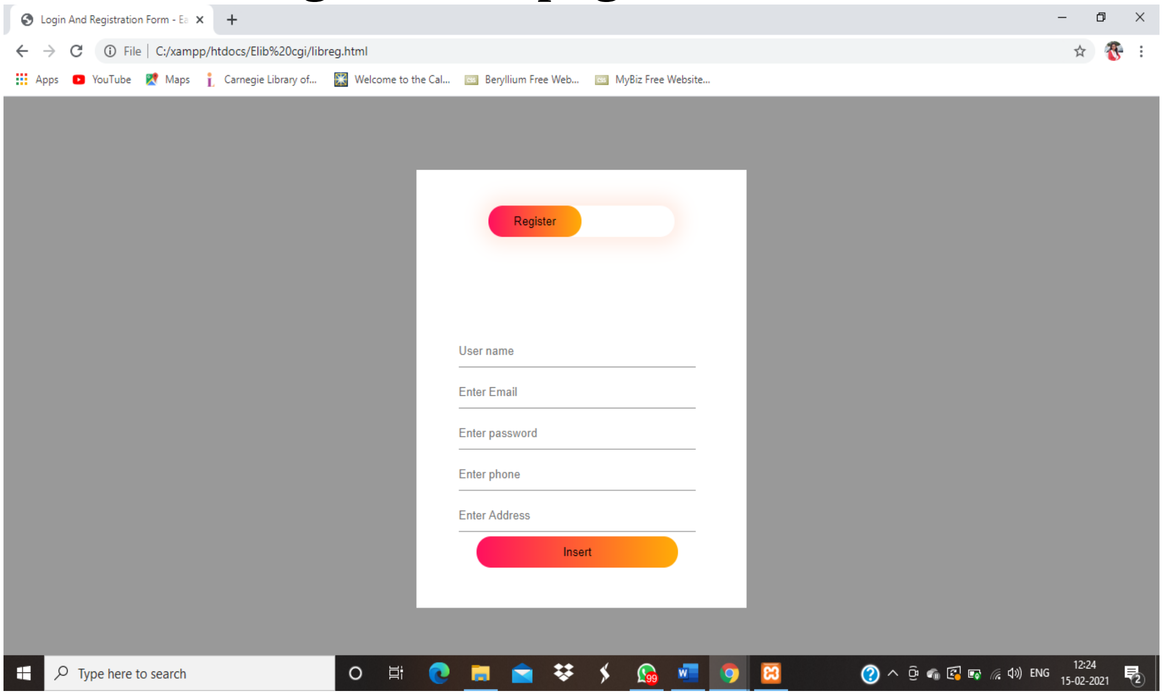
```
<html>

<head>

    <title>Login And Registration Form
- Easy Tutorials</title>

    <link rel="stylesheet"
href="css/style2.css">



</head>

<body>

    <div class="hero">

        <div class="form-box">

            <div class="button-box">

              <div id="btn"></div>

                <button type="button"
class="toggle-btn"
onclick="login()">Log In</button>
```

```
            </div>


        <form id="login" class="input-
group" action="adlog_code.py">
            <input type="text" class="input-
field" placeholder="User Id"
name="a1" required>
            <input type="password"
class="input-field" placeholder="Enter
password" name="a2" required>
            <input type="checkbox"
class="chech-box"><span>Remember
Password</span>
            <button type="submit"
class="submit-btn">Log in</button>


        </form>


    </div>
  </div>
  <script>
```

```
      var x =
document.getElementById("login");
      var y =
document.getElementById("register");
      var z =
document.getElementById("btn");


      function register(){
          x.style.left = "-400px";
          y.style.left = "50px";
          z.style.left = "110px";
      }
      function login(){
          x.style.left = "50px";
          y.style.left = "450px";
          z.style.left = "0";
      }
  </script>
</body>
```

```
</html>
```

Librarian Registration page



Librarian registration page code

```
<html>

<head>

   <title>Login And Registration Form - Easy Tutorials</title>

   <link rel="stylesheet" href="css/style2.css">




</head>

<body>

   <div class="hero">
```

```
        <div class="form-box">

            <div class="button-box">

              <div id="btn"></div>

                <button   type="button"   class="toggle-btn"
onclick="login()">Register </button>



            </div>



        <form         id="login"         class="input-group"
action="libreg_code.py">

            <input       type="text"       class="input-field"
placeholder="User name" name="a1" required>

            <input       type="text"       class="input-field"
placeholder="Enter Email" name="a2"required>

                  <input     type="text"     class="input-field"
placeholder="Enter password" name="a3"required>

                  <input     type="text"     class="input-field"
placeholder="Enter phone" name="a4" required>

                  <input     type="text"     class="input-field"
placeholder="Enter Address" name="a5" required>



            <button      type="submit"      class="submit-
btn">Insert</button>



        </form>
```

```
  </div>
 </div>
 <script>
   var x = document.getElementById("login");
   var y = document.getElementById("register");
   var z = document.getElementById("btn");

   function register(){
     x.style.left = "-400px";
     y.style.left = "50px";
     z.style.left = "110px";
   }
   function login(){
     x.style.left = "50px";
     y.style.left = "450px";
     z.style.left = "0";
   }
 </script>
</body>
</html>
```

IMPLEMENTATION AND TESTING

A software system test plan is a document that describes the objectives, scope, approach and focus of software testing effort. The process of preparing a test plan is a usual way to think the efforts needed to validate the acceptability of a software product. The complete document will help people outside the test group understand the "WHY" and "HOW" product validation. It should be through enough to be useful but not so through that no one outside the test group will read it.

6a. INTRODUCTION

Testing is the process of running a system with the intention of finding errors. Testing enhances the integrity of a system by detecting deviations in design and errors in the system. Testing aims at detecting error-prone areas. This helps in the prevention of errors in a system. Testing also adds value to the product by conforming to the user requirements.

The main purpose of testing is to detect errors and error-prone areas in a system. Testing must be thorough and well-planned. A partially tested system is as bad as an untested system. And the price of an untested and under-tested system is high.
The implementation is the final and important phase. It involves usertraining, system testing in order to ensure successful running of the proposed system. The user tests the system and changes are made according to their needs. The testing involves the testing of the developed system using various kinds of data. While testing, errors are noted and correctness is the mode.

6b. OBJECTIVES OF TESTING:

The objective our test plan is to find and report as many bugs as possible to improve the integrity of our program. Although exhaustive testing is not possible, we will exercise a broad range of tests to achieve our goal. Our user interface to utilize these functions is designed to be user-friendly and provide easy manipulation of the tree. The application will only be used as a demonstration tool, but we would like to ensure that it could be run from a variety of platforms with little impact on performance or usability.

Process Overview

The following represents the overall flow of the testing process:

1. Identify the requirements to be tested. All test cases shall be derived using the current Program Specification.

2. Identify which particular test(s) will be used to test each module.

3. Review the test data and test cases to ensure that the unit has been thoroughly verified and that the test data and test cases are adequate to verify proper operation of the unit.

4. Identify the expected results for each test.

5. Document the test case configuration, test data, and expected results.

6. Perform the test(s).

7. Document the test data, test cases, and test configuration used during the testing process. This information shall be submitted via the Unit/System Test Report (STR).

8. Successful unit testing is required before the unit is eligible for component integration/system testing.

9. Unsuccessful testing requires a Bug Report Form to be generated. This document shall describe the test case, the problem encountered, its possible cause, and the sequence of events that led to the problem. It shall be used as a basis for later technical analysis.

10. Test documents and reports shall be submitted. Any specifications to be reviewed, revised, or updated shall be handled immediately.

6c. TEST CASES

A test case is a document that describe an input, action, or event and expected response, to determine if a feature of an application is working correctly. A test case should contain particular such as test case identifier, test condition, input data
Requirement expected results. The process of developing test cases can help find problems in the requirement or design of an application, since it requires completely thinking through the operation of the application.

TESTING STEPS

Unit Testing:

Unit testing focuses efforts on the smallest unit of software design. This is known as module testing. The modules are tested separately. The test is carried out during programming stage itself. In this step, each module is found to be working satisfactory as regards to the expected output from the module.

Integration Testing:

Data can be lost across an interface. One module can have an adverse effect on another, sub functions, when combined, may not be linked in desired manner in major functions. Integration testing is a systematic approach for constructing the program structure, while at the same time conducting test to uncover errors associated within the interface. The objective is to take unit tested modules and builds program structure. All the modules are combined and tested as a whole.

Validation:

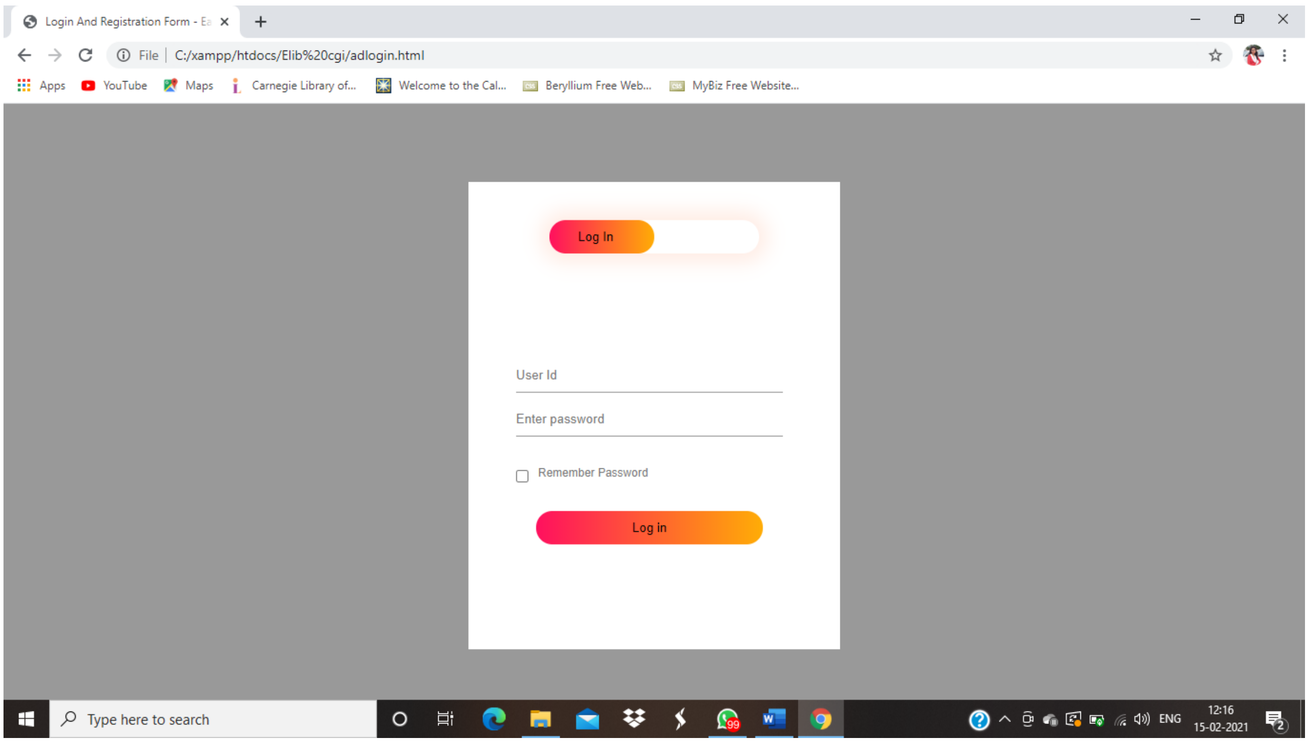
At the culmination of the integration testing, Software is completely assembled as a package. Interfacing errors have been uncovered and corrected and a final series of software test begin in validation testing. Validation testing can be defined in many ways, but a simple definition is that the validation succeeds when the software functions in a manner that is expected by the customer. After validation test has been conducted, one of the three possible conditions exists.

a)The function or performance characteristics confirm to specification and are accepted.

b)A deviation from specification is uncovered and a deficiency lists is created.
c)Proposed system under consideration has been tested by using validation test and found to be working satisfactory.

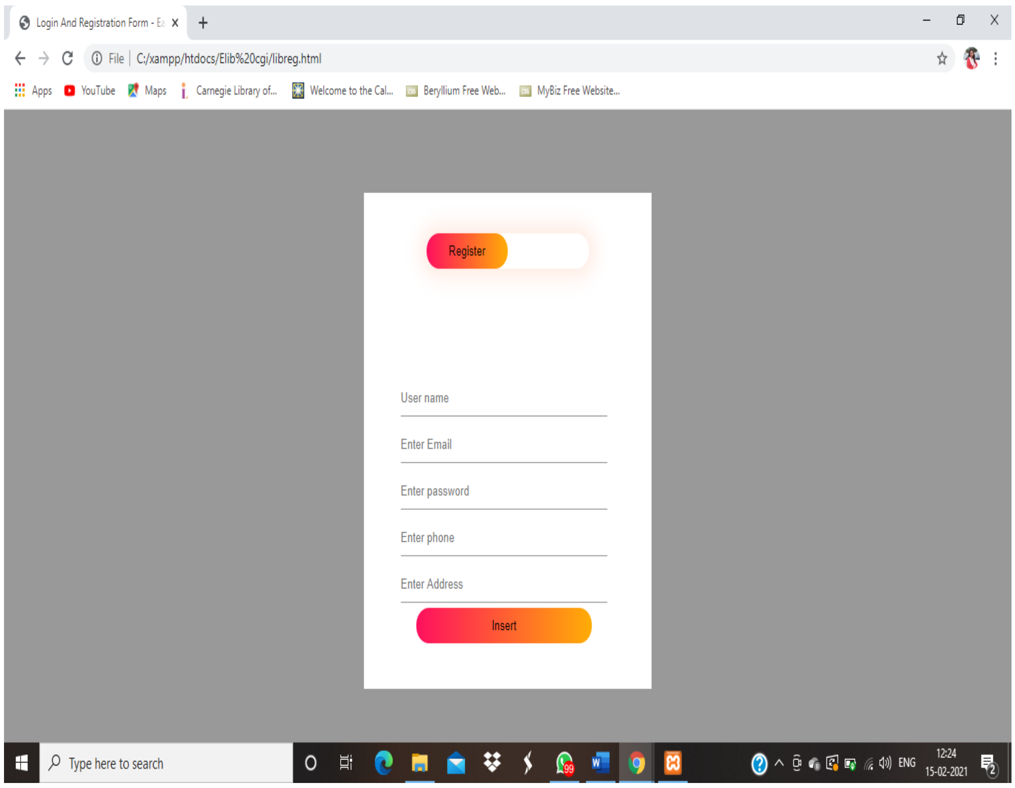
Tested By:	Suman Mondal
Test Type	Unit testing
Test Case Number	1
Test Case Name	Admin identification
Test Case Description	The admin should enter his/ her accurate admin id and password so that he/she can able to go for the further options. The test case will check the application for the same since a user can only login with the correct user id, password.
Item(s) to be tested	
1	Verification of the user id and password with the record in the database.
Specifications	
Input	**Expected Output/Result**
1) Correct User id and password 2) Incorrect Id or Password	1) Successful login 2) Failure Message

SNAPSHOT OF ADMIN LOGIN



Tested By:	Debasmita Shee	
Test Type	Unit testing	
Test Case Number	2	
Test Case Name	Registration of librarian	
Test Case Description	Admin will enter the details in the registration form and the credentials will be checked by the validators used in the form. After all the credentials are validated then only the form will be submitted and data will be stored in the database	
Item(s) to be tested		
1	Required fields in the form are not empty, validation of proper credentials	
Specifications		
Input	**Expected Output/Result**	
1) User id, name, password, phone, address, email. 2) Empty field, Invalid entry	1) Successful registration 2) Failure Message	

SNAPSHOTS OF LIBRARIAN REGISTRATION



6 d. WHITE BOX TESTING

In white box testing, the UI is bypassed. Inputs and outputs are tested directly at the code level and the results are compared against specifications. This form of testing ignores the function of the program under test and will focus only on its code and the structure of that code. Test case designers shall generate cases that not only cause each condition to take on all possible values at least once, but that cause each such condition to be executed at least once. To ensure this happens, we will be applying Branch Testing. Because the functionality of the program is relatively simple, this method will be feasible to apply.

Each function of the binary tree repository is executed independently; therefore, a program flow for each function has been derived from the code.

6e. BLACK BOX TESTING

Black box testing typically involves running through every possible input to verify that it results in the right outputs using the software as an end-user would. We have decided to perform Equivalence Partitioning and Boundary Value Analysis testing on our application.

System Testing

The goals of system testing are to detect faults that can only be exposed by testing the entire integrated system or some major part of it. Generally, system testing is mainly concerned with areas such as performance, security, validation, load/stress, and configuration sensitivity. But in our case well focus only on function validation and performance. And in both cases we will use the black-box method of testing.

6f. OUTPUT TESTING

After performing the validation testing, the next step is output testing of the proposed system, since no system could be useful if it does not produce the required output in a specific format. The output format on the screen is found to be correct. The format was designed in the system design time according to the user needs. For the hard copy also; the output comes as per the specified requirements by the user. Hence output testing did not result in any correction for the system.

User Acceptance Testing:

User acceptance of a system is the key factor for the success of any system. The system under consideration is tested for the user acceptance by constantly keeping in touch with the prospective system users at the time of developing and making changes whenever required.

This is done in regard to the following point: a) Input Screen Design
b) Output Screen Design
c) Format of reports and other outputs.

6g. GOAL OF TESTING

"Program testing can be used to slow the presence of bug, but never to slow their absence." If the results delivered by the system are different from the excepted ones then the system is incorrect and these bugs should be fixed.

6h. INTEGRATION TEST REPORTS

Software testing is always used in association with verification and validation. In the testing phase of this project our aim is to find the answer to following two questions.

- Whether the software matches with the specification (i.e. process base) to verify the product.

- Whether this software in one client what wants (i.e. product base) to validate the product.

  Unit testing and integration testing has been carried out to find the answer to above questions. In unit testing each individual module was test to find any unexpected behaviour if exists. Later all the module was integrated and flat file was generated.

FUNCTIONAL TESTING

These are the points concerned during the stress test:

- Nominal input: character is in putted in the place of digits and the system has to flash the message "Data error"
- Boundary value analysis: exhaustive test cases have designed to create an output report that produces the maximum (and minimum) allowable number of table entries.

Testing Method Used

We have adopted a testing method which is a mix of both (structural) and black box (functional) testing. For modules we have adopted white box testing. Then we integrated the module into sub - systems and further into the system. These we adopted black box testing for checking the correctness of the system.

Requirements Validated and Verified:

- The data is getting entered properly into database.
- The Screens are being loaded correctly
- The Various functions specified are being performed completely.

SYSTEM SECURITY MEASURES

8.a DATABASE SECURITY

System security measure is meant to be provided to make your system reliable and secured from unauthorized user may create threats to the system. So you should follow some security measures. We have used security levels in database level at system level.

8.b SYSTEM SECURITY

If we talk about the system security in our proposed system we have implemented with the help of maintain the session throughout the system's use. Once a user has logged out than he/she will not be able to perform any task before signing back again.

A high level of authentic login is given to the system so this is a very tedious task to enter without authorization and authentication.

8c. LIMITATIONS:

- ✓ Since it is an online project, customers need internet connection to use it.
- ✓ People who are not familiar with computers can’t use this software.
- ✓ Customer must have debit card or credit card to book tickets.

9. CONCLUSION

This project has been appreciated by all the users in the organization. It is easy to use, since it uses the CGI provided in the user dialog. User friendly screens are provided. The usage of software increases the efficiency, decreases the effort. It has been efficiently employed as a Site management mechanism. It has been thoroughly tested and implemented.

10. FUTURE SCOPE AND FURTHER ENHANCEMENTS

In future we would like to keep working on this project and make new additions to provide users with more advanced features and more detailed information. We have set our sights on the following additions in future:-

1. Forget Password for admin and councilor.
2. Online payment process through debit and credit cards.
3. Automail will be sent to the student's email-id when new admission takes places.

THANK YOU