EXPERIMENT NO: 01

Title : Apply suitable software development model for the given scenario.

Theory :

- **Software Development Process :**

  A software development process is the process of dividing software development work into distinct phases to improve design, product management, and project management. It is also known as a software development life cycle. Most modern development processes can be vaguely described as agile. Other methodologies include waterfall, prototyping, iterative and incremental development, spiral development and rapid application development.

- **Types of Development Models :**
  1. Waterfall Model.
  2. Incremental Model.
  3. RAD Model.
  4. Prototyping.
  5. Spiral Model.

- **Waterfall Model** :
  1. The Waterfall Model is earliest SDLC approach.
  2. It is also known as Classic Life Cycle Model.
  3. In this Model, Each phase must be completed before the next phase can begin.
- Features :
  1. It is good to use when technology is well understood.
  2. The project is short and cost is low.
  3. Risk is zero or simple and easy.

- Advantages :
  1. It is simple and easy.
  2. Easy to manage.
  3. It is good for low budget small project.

- Disadvantages :
    1. It is not good for complex and object oriented programming.
    2. It is a poor model for long and ongoing project.

- Diagram :

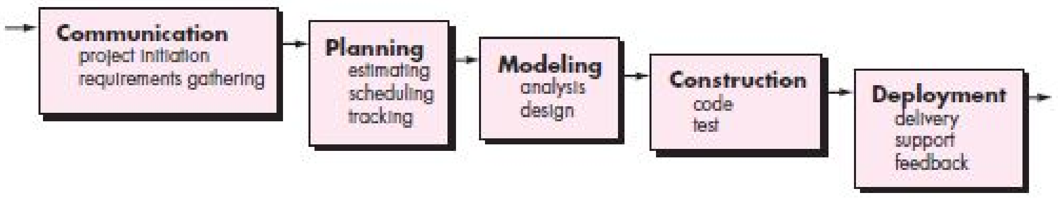


Fig.1.1 Waterfall Model

- **Incremental Model:**

Incremental model is a process of Software Development where requirements are broken down into multiple modules of Software Development Cycle. Increment means to add something. Incremental development is done in steps or in increments from communication, planning, modeling, construtions and deployment. Each iterations passes through all 5 phases . And each subsequent release of the system adds function to the previous relese until all designed fuctionality has been implemeted.

- Features :
    1. Incremental is the combination of Linear and Prototype.
    2. Incremental Delivery.
    3. Priority to user's highly recommended requirements.
    4. Involvement of Users.
    5. Lower risk.
    6. Highest Priority System means in each increment they can easily find errors.
    7. At small requirements we can start our model.
    8. After every cycle the product is given to the customer.

- Advantages:
    1. Work with small size team.
    2. Initial product delivery is faster.

3. Customer response or feedback is considered.
4. It is easier to test and Debug during a smaller iteration.
5. The model is more flexible.
6. Easier to manage risk.

- Disadvantages :
  1. Actual cost will be more than Estimated cost.
  2. Needs good planning and design.

- Diagram :

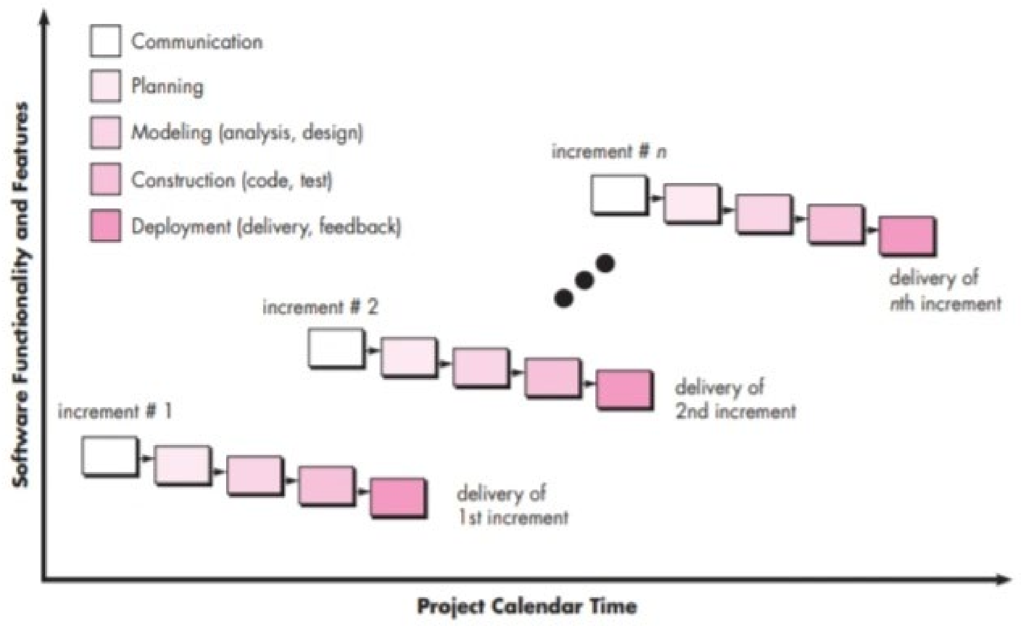

Communication
Planning
Modeling (analysis, design)
Construction (code, test)
Deployment (delivery, feedback)
increment # 1
delivery of 1st increment
increment # 2
delivery of 2nd increment
increment # n
delivery of nth increment
Software Functionality and Features
Project Calendar Time

Fig.1.2 Incremental Model.

- **RAD Model :**

RAD or Rapid Application Development process is an adoption of the waterfall model; it targets at developing software in a short span of time. RAD follow the iterative.

- SDLC RAD model has following phases
  1. Business Modelling.
  2. Data Modelling.
  3. Process Modelling.
  4. Application Generation.
  5. Testing and Turnover.

It focuses on input-output source and destination of the information. It emphasizes on delivering projects in small pieces; the larger projects are divided into a series of smaller projects. The main features of RAD model are that it focuses on the reuse of templates, tools, processes, and code.

- Different phases of RAD model includes :

- Business Modelling :

  On basis of the flow of information and distribution between various business channels, the product is designed.
- Data Modelling :

  The information collected from business modelling is refined into a set of data objects that are significant for the business.
- Process Modelling

  The data object that is declared in the data modelling phase is transformed to achieve the information flow necessary to implement a business function.
- Application Generation

  Automated tools are used for the construction of the software, to convert process and data models into prototypes.
- Testing and Turnover

  As prototypes are individually tested during every iteration, the overall testing time is reduced in RAD.

- When to use RAD Methodology?
  1. When a system needs to be produced in a short span of time (2-3 months).
  2. When the requirements are known.
  3. When the user will be involved all through the life cycle.
  4. When technical risk is less.
  5. When there is a necessity to create a system that can be modularized in 2-3 months of time.
  6. When a budget is high enough to afford designers for modelling along with the cost of automated tools for code generation.

- Advantages :
  1. Flexible and adaptable to changes.
  2. It is useful when you have to reduce the overall project risk.
  3. It is adaptable and flexible to changes.

4. It is easier to transfer deliverables as scripts, high-level abstractions and intermediate codes are used.
5. Due to code generators and code reuse, there is a reduction of manual coding.
6. Each phase in RAD delivers highest priority functionality to client.
7. With less people, productivity can be increased in short time.

- Disadvantages :
    1. It can't be used for smaller projects.
    2. Not all application is compatible with RAD.
    3. When technical risk is high, it is not suitable.
    4. If developers are not committed to delivering software on time, RAD projects can fail.
    5. Reduced features due to time boxing, where features are pushed to a later version to finish a release in short period.
    6. Reduced scalability occurs because a RAD developed application begins as a prototype and evolves into a finished application.
    7. Requires highly skilled designers or developers.

- Diagram :

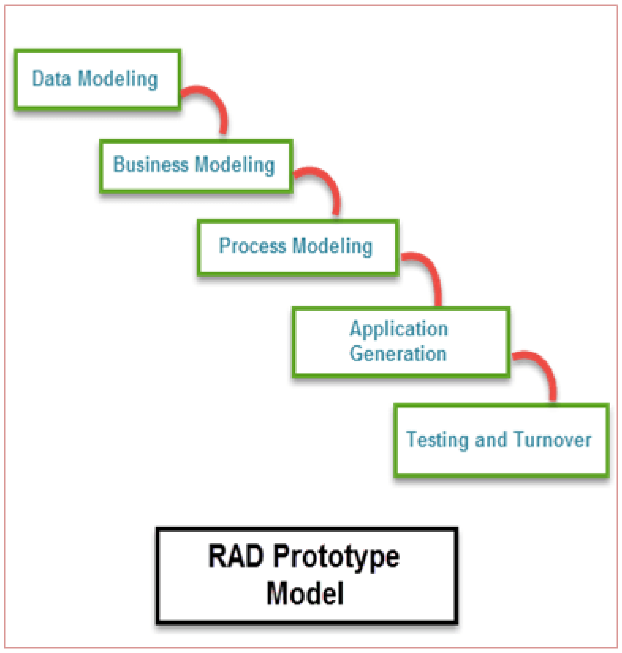


Fig.1.3 RAD Model.

- **Prototyping Model :**

  In this model, a prototype of a end product is first developed, tested and defined as per customer's feedback repeatedly till the final expectable prototype is achieved. The intension behind creating this model is to get actual requirements more deeply from the user.

- Process of Prototyping :
  1. Initial requirement identification.
  2. Prototype development.
  3. Review.
  4. Revise.

- Advantages :
  1. The customer get to see the partial product early in the life cycle.
  2. Missing functionality can be easily figured out.
  3. Flexibility in design.
  4. New requirements can be easily accommodated.

- Disadvantages :
  1. Costly with respect to time as well as money.
  2. Poor documentation due to continuously change in requirements.
  3. After seeing an early prototype the customer demands the actual product will be delivered soon.

- Diagram :

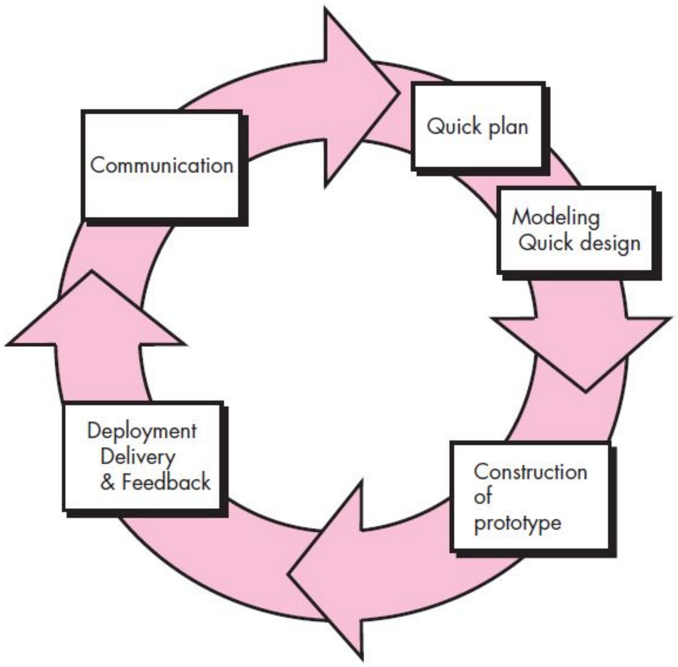


Fig.1.4 Prototype Model.

- **Spiral Model :**

The Spiral model was proposed by Barry Boehm. The Spiral model is the combination of Waterfall model and Incremental model. The Spiral model is divided into 4 task regions. Each task region begins with the design goal and ends with the client reviewing. Software is developed in the series of incremental release. The task region of Spiral model are :

1. Concept Development.
2. System Development.
3. System Enhancement.
4. System Maintenance.

- Features :
  1. When the release is frequent this model is used.
  2. Used for large projects.
  3. Compatible even if requirements are complex and unclear.
  4. Changes can be done at any time.
  5. When risk evaluation is important Spiral model is used.

- Advantages :
  1. Additional functions or changes can be done at later stage.
  2. Cost estimation becomes easy.
  3. There is always space for customer's feedback.

4. Development is fast and features are added in systematic way.

- Disadvantages :
  1. Documentation is more as it has intermediate phases.
  2. It is not advisable for smaller projects, as it may cost them a lot.

- Diagram :

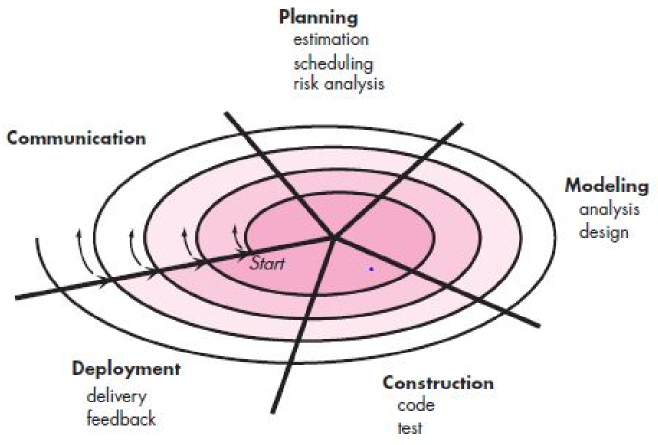


Fig. 1.5 Spiral Model.

- **Scenario: Development of employment salary attendance and biodata management system for a huge government company.**

  The salary attendance and biodata management system or application is one of the important requirements in enterprise or company. This system has many features like name of employees, their biodata, salary date, salary, etc and also after taking this data one can update the attendance. The previous data can also be fetched of any month or year.

- **Which model will you use?**

  According to me, for this salary attendance and biodata management application we may use prototyping or spiral model.

- **Why this model? Elaborate.**

  As we know the salary attendance and biodata management application very much complicated to design but also it should be use. In prototyping or a dummy model of the software is made and presented to the customers. According to their requirement they will ask to modify or keep it same. Similarly, in spiral the project is completed in the form of task region so that the requirements of the customers are easily known and any modification can be done even in later stage. So, the spiral model decreases the complexity to design. Thus, in this way we can implement both models.

Conclusion:

Thus, in this practical I learnt about 5 different types of development model in detail. And also solved questions related to scenario of attendance application.

(10)	(20)	(10)	(10)	Total(50)

EXPERIMENT NO: 02

Title: Identify the objectives and summarize outcomes for given scenario, for each SDLC phase.

Theory:

- **Software Development Life Cycle:**

  A software development process life cycle (SDLC) is a framework defining tasks performed at each step in the software development process. SDLC is a structure followed by a development team within the software organization. It consists of a detailed plan describing how to develop, maintain and replace specific software. The life cycle defines a methodology for improving the quality of software and the overall development process.

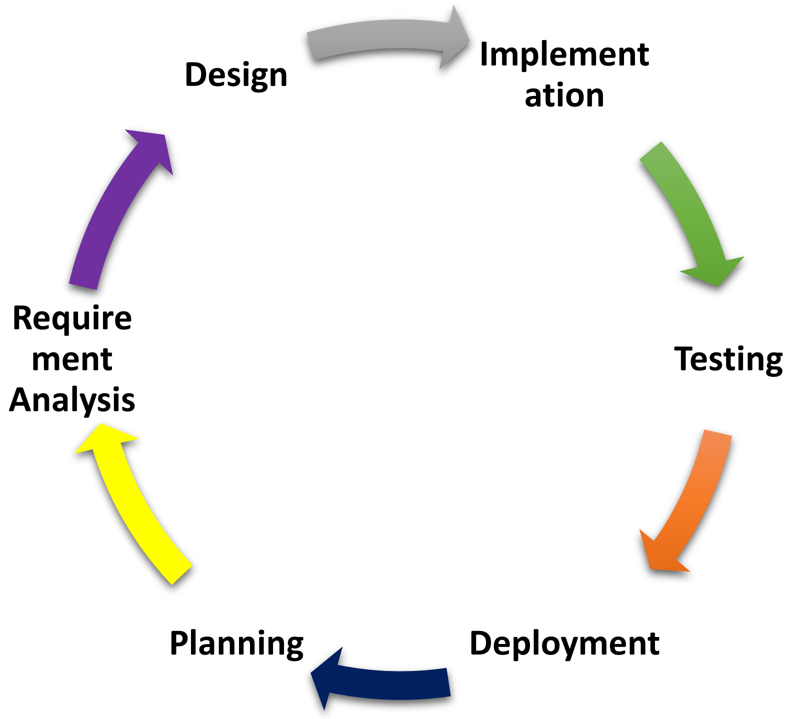


Fig: SDLC phases

- **SDLC Phases**

  1. Planning
  2. Requirement Analysis
  3. Design

4. Implementation
5. Testing
6. Deployment and Maintenance

- **Planning**

The first step in the SDLC defines the scope of the project. For example, your sales reps might be stuck with a hodgepodge of apps on Androids and iPhones to track sales leads. You, the owner, want them on a unified system that works better with your company's internal software. This may mean changing to a single type of hardware and choosing sales tracking software managed by the home office. System planning is the process of deciding what your new information system should look like and then identifying the resources needed to develop it.

- **Requirement Analysis**

1. **Communication**

   Another Part of Requirement Analysis is Communication where the stakeholders discuss the requirements of the software that needs to be developed to achieve a goal. The aim of the requirement analysis phase is to capture the detail of each requirement and to make sure everyone understands the scope of the work and how, spiral development and each requirement is going to be fulfilled

2. **Requirement Gathering**

   Requirements Gathering (also known as Requirements elicitation or Capture) is the process of generating a list of requirements (functional, system, technical, etc.) from the various stakeholders (customers, users, vendors, IT staff, etc.) that will be used as the basis for the formal Requirement Definition. The process is not as straightforward as just asking the stakeholders what they want they system to do, as in many cases, they are not aware of all the possibilities that exist, and may be limited by their immersion in the current.

3. **Defining Requirement**

Ones the requirement analysis is done next step is to delay defining the documents the product requirement and get them approved from the customer or the market analysts. This is done through an SRS document which is consist of all the product requirement to be designed and deployed during the project life cycle.

It Also Consist: 1.Cost Estimation

- **Design and Functionality**
  Functionality
  Step three in the SDLC is reserved for listing features that support the system's proper functioning. For example, an inventory control system may need to handle at least 15 users or that it should interface with the U.S. Customs database as a compliance check on imports.

  1.Tracking

  2.Scheduling

  Design

  1. Algorithm
  2. Flowchart

- During the design phase, developers and technical architects start the high-level design of the software and system to be able to deliver each requirement.
- The technical details of the design is discussed with the stakeholders and various parameters such as risks, technologies to be used, capability of the team, project constraints, time and budget are reviewed and then the best design approach is selected for the product.
- The selected architectural design, defines all the components that needs to be developed, communications with third party services, user flows and database communications as well as front-end representations and behavior of each components. The design is usually kept in the Design Specification Document (DSD)

- **Implementation/Coding**

After the requirements and design activity is completed, the next phase of the Software Development Life Cycle is the implementation or development of the software. In this phase, developers start coding according to the requirements and the design discussed in previous phases. Database admits create the necessary data in the database, front-end developers create the necessary interfaces and GUI to interact with the back-end all based on guidelines and procedures defined by the company.

- **Testing**

After the code is developed it is tested against the requirements to make sure that the product is actually solving the needs addressed and gathered during the requirements phase. During this phase all types of functional testing like unit testing, integration testing, system testing, acceptance testing are done as well as non-functional testing are also done.

This cycle is repeated until all requirements have been tested and all the defects have been fixed and the software is ready to be shipped.

- **Deployment and Maintenance**

Once the software has been fully tested and no **high priority issues** remain in the software, it is time to deploy to production where customers can use the system. Once a version of the software is released to production, there is usually a maintenance team that look after any post-production issues .If an issue is encountered in the production the development team is informed and depending on how severe the issue is, it might either require a hot-fix which is created and shipped in a short period of time or if not very severe, it can wait until the next version of the software.

- **Scenario: Online communication/ chatting web application**
  The online communication or chatting web application is one of the most vastly spreaded software and which is very continent to use and it very much easily adaptable. The online chatting application include sender, receiver, message, attachments, etc. The online communication system reduced a lot of problem arises in the functionality if communication.

I. Planning

Online communication/ chatting web application

In the planning phase the project planning is done i.e., estimation, scheduling and tracking is done.

- In Estimation the whole budget of the project is made including the salary of everyone, resources, raw materials, etc.
- Scheduling, in this the time required for the project is calculated for proper time management.
- Tracing in this different no of milestones is made for proper investigation and proper working

II. Requirement Analysis

Online communication/ chatting web application

In this phase the requirements from the customer are gathered and according to which the project is constructed. The requirement analysis must be done properly for proper development of the project. The requirement must be gathered and then analyzed and confirmed it from the customer.

III. **Modelling**

Online communication/ chatting web application

In modelling phase the flowchart and the algorithm are made the flowcharts are made for proper work flow of the project. The algorithms and flowcharts help the developer in the development of the project.

IV. **Construction**

Online communication/ chatting web application

In construction phase the real development of the project starts. After proper planning gathering of requirement modelling the construction is done.

V. **Deployment**

Online communication/ chatting web application

After the construction and approval of the software the software is deployed.

The deployment consists of delivery, feedback and support.

1. Delivery means proper installation of software.
2. After a specific period of time, the feedback is taken from the users.
3. And after feedback according to it support is provided.

Questions:

- **Which process according to you works simultaneously and Why?**

1. Modelling and construction:
   Because the project is designed in the form of module if the modelling of single module started and modelling of others modules is done simultaneously.

2. Communication and planning:

After the communication, planning done but it can be done simultaneously as requirements gathered as well as planning can be done. It reduces the time.

- **Which of the following is the most important phase in SDLC? why?**

  All the phase in SDLC is important because everyone is interdependent on each other. Do no single phase is an important phase in SDLC.

- **From where do defects and failures in software testing arise?**

  The defects and failures may arise from the requirement gathering and also may arise from no proper modelling as it has flowcharts and algorithms it must be proper. The construction phase can also may arise defects because the real time coding is done.

- **Which phase of the SDLC is known as the "ongoing phase" and why?**

According to my view, I think every phase except requirement gathering is an ongoing phase because whenever updates or support is required all the phase must be done.

- **Conclusion**

  Thus, In this practical I identified the SDLC phases learnt in detail about each phase and scenario based on Online communication/ chatting web application and learnt about each phase regarding to Online communication/ chatting web application

(10)	(20)	(10)	(10)	Total(50)

EXPERIMENT NO: 03

Title : Design Software Requirement Specification (SRS) document for the project. Consider any project to be developed in any technology as a software Architect or project Manager.

Theory:

What is a Software Requirements Specification?

A software requirements specification is a document which is used as a communication medium between the customer and the supplier. When the software requirement specification is completed and is accepted by all parties, the end of the requirements engineering phase has been reached. This is not to say, that after the acceptance phase, any of the requirements cannot be changed, but the changes must be tightly controlled. The software requirement specification should be edited by both the customer and the supplier, as initially neither has both the knowledge of what is required (the supplier) and what is feasible (the customer).

Why is a Software Requirement Specification Required

A software requirements specification has a number of purposes and contexts in which it is used. This can range from a company publishing a software requirement specification to companies for competitive tendering, or a company writing their own software requirement specification in response to a user requirement document. In the first case, the author of the document has to write the document in such a way that it is general enough as to allow a number of different suppliers to propose solutions, but at the same time containing any constraints which must be applied. In the second instance, the software requirement specification is used to capture the users requirements and if any, highlight any inconsistencies and conflicting requirements and define system and acceptance testing activities.

A software requirement specification in its most basic form is a formal document used in communicating the software requirements between the customer and the developer. With this in mind then the minimum amount of information that the software requirement specification should contain is a list of requirements which has been agreed by both parties. The requirements, to fully satisfy the user should have the However the requirements will only give a narrow view of the system, so more information is required to place the system into a context which defines the purpose of the system, an overview of the systems functions and the type of user that the system will have. This additional information will aid the developer in creating a software system which will be aimed at the user’s ability and the client’s function.

Types of Requirements

Whilst requirements are being collated and analysed, they are segregated into type categories. The European Space Agency defined possible categories as

- Functional requirements,
- Performance requirements,
- Interface requirements,
- Operational requirements,
- Resource requirements,
- Verification requirements,
- Acceptance testing requirements,
- Documentation requirements,
- Quality requirements,
- Safety requirements,
- Reliability requirements and
- Maintainability requirements

Functional Requirements

Functional or behavioural requirements are a sub-set of the overall system requirements. These requirements are usedto consider trade-offs, system behaviour, redundancy and human aspects. Trade-offs may be between hardware and software issues, weighing up the benefits of each. Behavioural requirements, as well as describing how the system will operate under normal operation should also consider the consequences and response due to software failure or invalid inputs to the system.

Performance Requirements

All performance requirements must have a value which is measurable and quantitative, not a value which is perceptive.Performance requirements are stated in measurable values, such as rate, frequency, speeds and levels. The values specified must also be in some recognised unit, for example metres, centimetre square, BAR, kilometres per hour, etc.The performance values are based either on values extracted from the system specification, or on an estimated value.

Interface Requirements

Interface requirements, at this stage are handled separately, with hardware requirements being derived separately from the software requirements. Software interfaces include dealing with an existing software system, or any interface standard that has been requested. Hardware requirements, unlike software give room for trade-offs if they are not fully defined, however all assumptions should be defined and carefully documented.

Operational Requirements

Operational requirements give an "in the field" view to the specification, detailing such things as:

- how the system will operate,
- what is the operator syntax,
- how the system will communicate with the operators,
- how many operators are required and their qualification,
- what tasks will each operator be required to perform,
- what assistance/help is provided by the system,

 any error messages and how they are displayed, and

what the screen layout looks like.

Resource Requirements

Resource requirements divulge the design constraints relating to the utilisation of the system hardware. Software restrictions may be placed on only using specific, certified, standard compilers and databases. Hardware restrictions include amount, percentage or mean use of the available memory and the amount of memory available. The definition of available hardware is specially important when the extension of the hardware, late in the development life cycle is impossible or expensive.

Verification Requirements

Verification requirements take into account how customer acceptance will be conducted at the completion of the project. Here a reference should be made to the verification plan document. Verification requirements specify how the functional and the performance requirements are to be measured and verified. The measurements taken may include simulation, emulation and live tests with real or simulated inputs. The requirements should also state whether the measurement tests are to be staged or completed on conclusion of the project, and whether a representative from the clients company should be present.

Acceptance Testing Requirements

Acceptance test requirements detail the types of tests which are to be performed prior to customer acceptance. These tests should be formalised in an acceptance test document.

Documentation Requirements

Documentation requirements specify what documentation is to be supplied to the client, either through or at the end of the project. The documentation supplied to the client may include project specific documentation as well as user guides and any other relevant documentation.

Quality Requirements

Quality requirements will specify any international as well as local standards which should be adhered to. The quality requirements should be addressed in the quality assurance plan, which is a core part of the quality assurance document.

Typical quality requirements include following ISO9000-3 procedures . The National Aeronautics and Space Administrations software requirement specification - SFW-DID-08 goes to the extent of having subsections

detailing relevant quality criteria and how they will be met. These sections are Quality Factors :

- Correctness
- Reliability
- Efficiency
- Integrity
- Usability
- Maintainability
- Testability
- Flexibility
- Portability
- Reusability
- Interoperability
- Additional Factors

Some of these factors can be addressed directly by requirements, for example, reliability can be stated as an average period of operation before failure. However most of the factors detailed above are subjective and may only be realised during operation or post delivery maintenance. For example, the system may be vigorously tested, but it is not always possible to test all permutations of possible inputs and operating conditions. For this reason errors may be found in the delivered system. With correctness the subjectiveness of how correct the system is, is still open to interpretation and needs to be put into context with the overall system and its intended usage. An example of this can be taken from the recently publicised 15th point rounding error found in Pentium(processors. In the whole most users of the processor will not be interested in values of that order, so as far as they are concerned, the processor meets their correctness quality criteria, however a laboratory assistant performing minute calculations for an experiment this level or error may mean that the processor does not have the required quality of correctness.

Safety Requirements

Safety requirements cover not only human safety, but also equipment and data safety. Human safety considerations include protecting the operator from moving parts, electrical circuitry and other physical dangers. There may be special operating procedures, which if ignored my lead to a hazardous or dangerous condition occurring. Equipment safety includes safeguarding the software system from unauthorised access either electronically or physically. An example of a safety requirement may be that a monitor used in the system will conform to certain screen emission standards or that the system will be installed in a Faraday Cage with a combination door lock.

Reliability Requirements

Reliability requirements are those which the software must meet in order to perform a specific function under certain stated conditions, for a given period of time. The level of reliability requirement can be dependant on the type of system, i.e. the more critical or life threatening the system, the higher the level of reliability required. Reliability can be measured in a number of ways including number of bugs per x lines of code, mean time to failure and as a percentage of the time the system will be operational before crashing or an error occurring. Davis states however that the mean time to failure and percent reliability should not be an issue as if the software is fully tested, the error will either show itself during the initial period of use, if the system is asked to perform a function it was not designed to do or the hardware/software configuration of the software host has been changed . Davis suggests the following hierarchy when considering the detail of reliability in a software requirement specification.

- Destroy all humankind
- Destroy large numbers of human beings
- Kill a few people
- Injure people
- Cause major financial loss
- Cause major embarrassment
- Cause minor financial loss

- Cause mild inconvenience
- Naming conventions

- Component headers
- In-line document style
- Control constructs
- Use of global/common variables

Maintainability Requirements

Maintainability requirements look at the long term lift of the proposed system. Requirements should take into consideration any expected changes in the software system, any changes of the computer hardware configuration and special consideration should be given to software operating at sites where software support is not available. Davis suggests defining or setting a minimum standard for requirements which will aid maintainability i.e.

Characteristics of a Good Software Requirements Specification

A software requirements specification should be clear, concise, consistent and unambiguous. It must correctly specify all of the software requirements, but no more. However the software requirement specification should not describe any of the design or verification aspects, except where constrained by any of the stakeholders requirements.

Complete

For a software requirements specification to be complete, it must have the following properties:

Description of all major requirements relating to functionality, performance, design constraints and external interfaces.

Definition of the response of the software system to all reasonable situations.

Conformity to any software standards, detailing any sections which are not appropriate

Have full labelling and references of all tables and references, definitions of all terms and units of measure.

Be fully defined, if there are sections in the software requirements specification still to be defined, the software requirements specification is not complete

Consistent

A software requirement specification is consistent if none of the requirements conflict. There are a number of different

types of confliction:

Multiple descriptors - This is where two or more words are used to reference the same item, i.e. where the term cue and prompt are used interchangeably.

Opposing physical requirements - This is where the description of real world objects clash, e.g. one requirement states that the warning indicator is orange, and another states that the indicator is red.

Opposing functional requirements - This is where functional characteristics conflict, e.g. perform function X after both A and B has occurred, or perform function X after A or B has occurred.

Traceable

A software requirement specification is traceable if both the origins and the references of the requirements are available. Traceability of the origin or a requirement can help understand who asked for the requirement and also what modifications have been made to the requirement to bring the requirement to its current state. Traceability of references are used to aid the modification of future documents by stating where a requirement has been referenced. By having foreword traceability, consistency can be more easily contained.

Unambiguous

As the Oxford English dictionary states the word unambiguous means "not having two or more possible meanings". This means that each requirement can have one and only one interpretation. If it is unavoidable to use an ambiguous term in the requirements specification, then there should be clarification text describing the context of the term. One way of removing ambiguity is to use a formal requirements specification language. The advantage to using a formal language is the relative ease of detecting errors by using lexical syntactic analysers to detect ambiguity. The disadvantage of using a formal requirements specification language is the learning time and loss of understanding of the system by the client.

Verifiable

A software requirement specification is verifiable if all of the requirements contained within the specification are verifiable. A requirement is verifiable if there exists a finite cost-effective method by which a person or machine can check that the software product meets the requirement. Non-verifiable requirements include "The system should have a good user interface" or "the software must work well under most conditions" because the performance words of good, well and most are subjective and open to interpretation. If a method cannot be devised to determine whether the software meets a requirement, then the requirement should be removed or revised.

- **SCENARIO:**

Online Pharmacy System

Online stores usually enable shoppers to user “search” features to find specific medicine, healthcare products or diagnosis tests. Online customer must have access to the internet and a valid method of payment in order to complete the transaction, such as credit card, an debit card, or service such as paytm, google pay or other UPI based transaction. Consumers find a product of interest by visiting the website of the retailer directly or by searching among alternative vendors using a shopping search engine. Once a particular product has been found on the website of the seller, most online retailers use shopping cart software to allow the consumers to accumulate multiple items and adjust quantities, like filling a physical shopping cart or basket in a conventional store.

Introduction:

- **Purpose**
  The online pharmacy store web application is intended to provide complete solutions for vendors as well as customers through a single gateway using the internet as the sole medium. It will enable Vendors to setup online pharmacy store, customer to browse through the shop and purchase them online without having to visit shop physically.

  This document is meant to discuss the features of online computer store, so as to serve as a guide to the developers on one hand and a software validation document for the perspective client on the other. The administration module will enable a system administrator to approve and reject requests for new shops and maintain various list of shop category.

- **Audience Definition**
  The intended readers of this document are developers of the site, testers, website owners, managers and coordinators.

- **Acronyms & Abbreviations**
  OPS: Online pharmacy Store
  HTTP: Hyper Text Transfer Protocol
  TCP/IP: Transmission Control Protocol / Internet Protocol

- **Technologies to be used**
  Java EE
  HTML, XML
  JavaScript
  Apache tomcat 7.8 server
  Eclipse J2EE

OVERALL DESCRIPTION:

- **Product Perspective**

  OPS is aimed towards the vendors who want to reach out to the maximum cross section of customer and common people who can be potential customer. This project envisages bridging the gap between the seller, retailers and the customer.

- **Product Function**

  User: Administration
  Function: The administrator is the super user and has complete control over all the activities that can be performed.

  User: Shop Owner
  Function: Any user can submit a shop creation request through the application.

  User: Customer/Guests
  Function: A customer can browse through the shops and choose products to place in a virtual shopping cart.

  User: Employees
  Function: Purchase department under a purchase manager to overlook purchasing activities if warehouse need arise.

SPECIFICATION REQUIREMENT

Function requirements:

1. **Administrator**
    a. Database management
    b. Contact and giving permission to vendors
    c. View all details
    d. Advertising the site
2. **Customers**
    a. Login
    b. Registration
    c. View and edit own details
    d. Purchasing
    e. Giving feedback to customer care
    f. Logout
3. **Visitors**
    a. Visiting the site
    b. Register
4. **Shop owner**
    a. Taking permission from administrator
    b. Consulting with administrator
    c. Advertising vendors own product
5. **Sales manager**
    a. View customer details
    b. Managing sales to customers
    c. View product stocks
    d. Contacting with administrator

Non-Functional Requirements

1. Performance requirement
2. Safety requirement
3. Security requirements
4. Error handling

Appendices:

www.guru99.com

www.wikipedia.com

www.google.com

Index:

- Introduction
- Overall Description
- Specification Requirement
- Appendices

CONCLUSION:

Thus, in this practical I learnt about SRS (software Requirement Specification) SRS components like Introduction, overall description, specification requirement, etc and also created an SRS for online pharmacy store.

(10)	**(20)**	**(10)**	**(10)**	**TOTAL (50)**

EXPERIMENT NO: 04

Title: Classify above identified requirement into functional and non-functional requirements.

Theory:

 What is SRS?

SRS stands for Software Requirement Specification. Software requirement is a functional or non-functional need to be implemented in the system. Functional means providing particular service to the user.
For example, in context to banking application the functional requirement will be when customer select “View Balance” they must be able to look at their latest account balance.
Software requirement can also be a non-functional, it can be a performance requirement. For example, a non-functional requirement is where every page of the system should be visible to the users within 5 seconds.
So, basically Software requirement is a

 Functional or

 Non-functional

need that has to be implemented into the system. Software requirement are usually expressed as a statements.
1

 Functional Requirement:

Functional requirements are the desired operations of program, or system as defined in software development and systems engineering. It describes the functions a software must perform. A function is nothing but inputs, its behaviour, and outputs. It can be a calculation, data manipulation, business process, user interaction, or any other functionality which defines what function a system is likely to perform. Functional Requirements are also called Functional Specification.

 Non-Functional Requirement:

Non-functional requirements describe how the system works. A non-functional requirement defines the quality attribute of a software system. They represent a set of standards used to judge the specific operation of system. A non-functional requirements is essential to ensure the usability and effectiveness of the entire software system. Non-functional requirements are often called Quality attributes of a system. Example, how fast does the website load?

 Scenario: **Development of employment salary attendance and biodata management system for a huge government company**

Employment Salary Attendance and Biodata Management System

The salary attendance and biodata management system or application is one of the important requirements in enterprise or company. This system has many features like name of employees, their biodata, salary date, salary, etc and also after taking this data one can update the attendance. The previous data can also be fetched of any month or year.

 What are the Functional Requirements for above scenario?

Function requirements:

1. **Administrator**
   a. **Database management**: control the database and keep the track of all records of employee details.

   b. **View all details:** View the details of all employees and control the whole site.

   c. **Calculation:** responsible for calculating the salary.

2. **Employees**
   a. **Login:** Customer must have a valid login id to enter into the site.

   b. **Registration:** New users can sign up by creating new ID.

   c. **View and edit own details:** can view/edit his personal details, payment details and details about services provided.

d. **Attendance:** can put attendance.

e. **Biodata:** put their biodata in the website.

f. **Logout:** employee must logout of the site after attendance.

 What are the Non-functional Requirements for above scenario?

Non-Functional Requirements

1. **Performance requirement:** The system shall accommodate high number of items and users without any fault.
   Response to view information shall take no longer than 5 seconds to appear on the screen.

2. **Safety requirement:** System use shall not cause any harm to human users.

3. **Security requirements:** System will use secured database.
   Normal users can just read information but they cannot edit or modify anything except their personal and some other information
   System will have different types of users and every user has access constraints.

4. **Error handling:** shall handle expected and non-expected errors in was that prevent loss in information and long downtime period.

** What are Benefits of Non-functional Requirements?**

- The non-functional requirement ensures the software follow legal and compliance rules.
- They ensure the reliability, availability and performance of the software system.
- They ensure good user experience and ease of operating the software.
- They help in formulating security policy of the software system.

□ Differentiate between Functional Requirements and Non-functional Requirements?

Functional Requirement	**Non- Functional Requirement**
It is mandatory.	It is not mandatory.
It is captured in use case.	It is captured in as a quality attribute.
Product feature	Product properties
Easy to capture	Hard to capture
Helps you verify the functionality of the software	Helps you to verify the performance of software
Focus on user's requirement	Concentrates on the users expectations
Describe what the product does	Describe how the product works
Functional testing like system. Integration, end to end, API testing, etc	Non-functional testing like performance, stress, usability, security testing, etc
Test execution is done before non-functional testing	After the functional testing

□ CONCLUSION:

Thus, in this practical I learnt about the functional and nonfunction requirements its advantages, disadvantages and also the functional and non-functional requirements for employment salary attendance and biodata management system.

(10)	**(20)**	**(10)**	**(10)**	**TOTAL (50)**

EXPERIMENT NO: 05

Aim: Design USE case diagram, state diagram for given scenario.

Theory:

What is a USE case diagram?

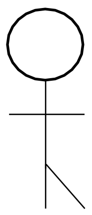
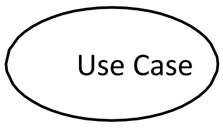
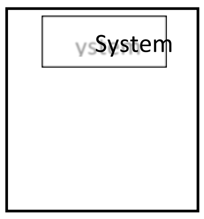
In the Unified Modeling Language (UML), a use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system. To build one, you'll use a set of specialized symbols and connectors. An effective use case diagram can help your team discuss and represent:

Scenarios in which your system or application interacts with people, organizations, or external systems.
Goals that your system or application helps those entities (known as actors) achieve.
The scope of your system.

When to apply use case diagrams

Use case diagrams specify the events of a system and their flows. But use case diagram never describes how they are implemented. Use case diagram can be imagined as a black box where only the input, output, and the function of the black box is known.

These diagrams are used at a very high level of design. This high- level design is refined again and again to get a complete and practical picture of the system. A well-structured use case also describes the pre-condition, post condition, and exceptions. These extra elements are used to make test cases when performing the testing.
Use case diagrams can be used for –
Requirement analysis and high-level design.
Model the context of a system.
Reverse engineering.
Forward engineering

Notation Description	Visual Representation
Actor Someone interacts with use case (system function). Similar to the concept of user, but a user can play different roles Actor has a responsibility toward the system (inputs), and Actor has expectations from the system (outputs).	
Use Case System function (process - automated or manual) i.e. Do something Each Actor must be linked to a use case, while some use cases may not be linked to actors.	
Communication Link Actors may be connected to use cases by associations, indicating that the actor and the use case communicate with one another using messages.	
Boundary of system The system boundary is potentially the entire system as defined in the requirements document. For example, for an ERP system for an organization, each of the modules such as personnel, payroll, accounting, etc.	

Relationships in Use Case Diagrams
There are five types of relationships in a use case diagram. They are

Association between an actor and a use case
Generalization of an actor
Extend relationship between two use cases
Include relationship between two use cases
Generalization of a use case

How to Create a Use Case Diagram

Identifying Actors

Actors are external entities that interact with your system. It can be a person, another system or an organization. In a banking system, the most obvious actor is the customer. Other actors can be bank employee or cashier depending on the role you're trying to show in the use case.

Identifying Use Cases

Now it's time to identify the use cases. A good way to do this is to identify what the actors need from the system. In a banking system, a customer will need to open accounts, deposit and withdraw funds, request check books and similar functions. So all of these can be considered as use cases.

Top level use cases should always provide a complete function required by an actor. You can extend or include use cases depending on the complexity of the system.

Look for Common Functionality to use Include

Look for common functionality that can be reused across the system. If you find two or more use cases that share common functionality you can extract the common functions and add it to a separate use case. Then you can connect it via the include relationship to show that it's always called when the original use case is executed.

Is it Possible to Generalize Actors and Use Cases

There may be instances where actors are associated with similar use cases while triggering a few use cases unique only to them. In such instances, you can generalize the actor to show the inheritance of functions.

One of the best examples of this is "Make Payment" use case in a payment system. You can further generalize it to "Pay by Credit Card", "Pay by Cash", "Pay by Check" etc. All of them have the attributes and the functionality of payment with special scenarios unique to them.

Optional Functions or Additional Functions

There are some functions that are triggered optionally. In such cases, you can use the extend relationship and attach an extension rule to it. In the below banking system example "Calculate Bonus" is optional and only triggers when a certain condition is matched.

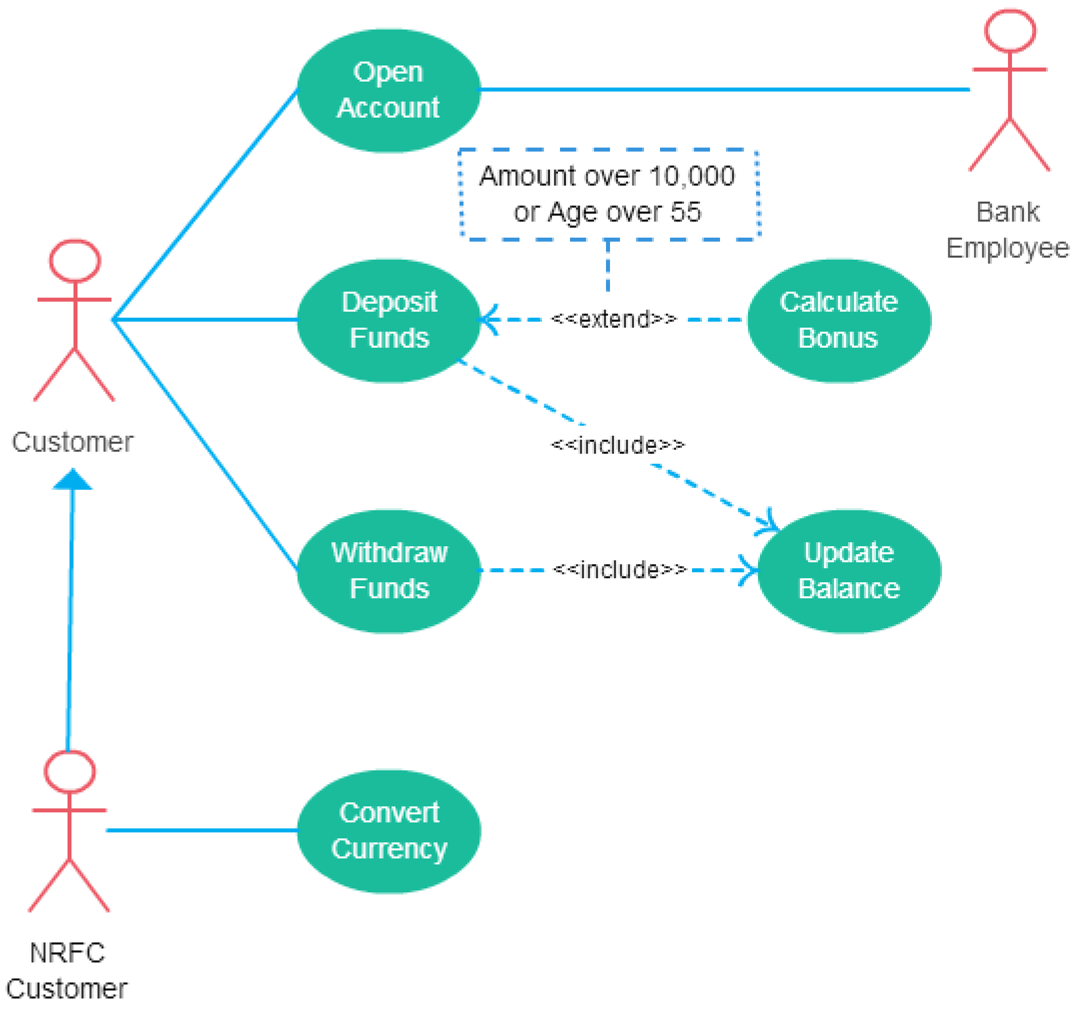


Fig. Example of use case diagrams

- **Scenario: Online Banking System**

Online Banking System is a major innovation in the field of Banking. Banking is now no longer confined to the branches where one has to approach the branch in person, to withdraw cash or deposit a cheque or request a statement of accounts. In true Internet banking, any inquiry or transaction is processed online without any reference to the branch (anywhere banking) at any time. Providing Internet banking is increasingly becoming a "need to have" than a "nice to have" service.

Questions:

What is the Importance of Use Case Diagrams?

- Use cases are important because they are in a tracking format.
- They make it easy to comprehend about the functional requirement in the system and also make it easy to identify the various interaction between the users and the systems within an environment.

- They are descriptive and hence clearly represent the value of an interaction between actors and the system.
- They clarify system requirements very categorically and systematic making it easier to understand the system and its interactions with the users.

Draw a USE case diagram for given scenario?

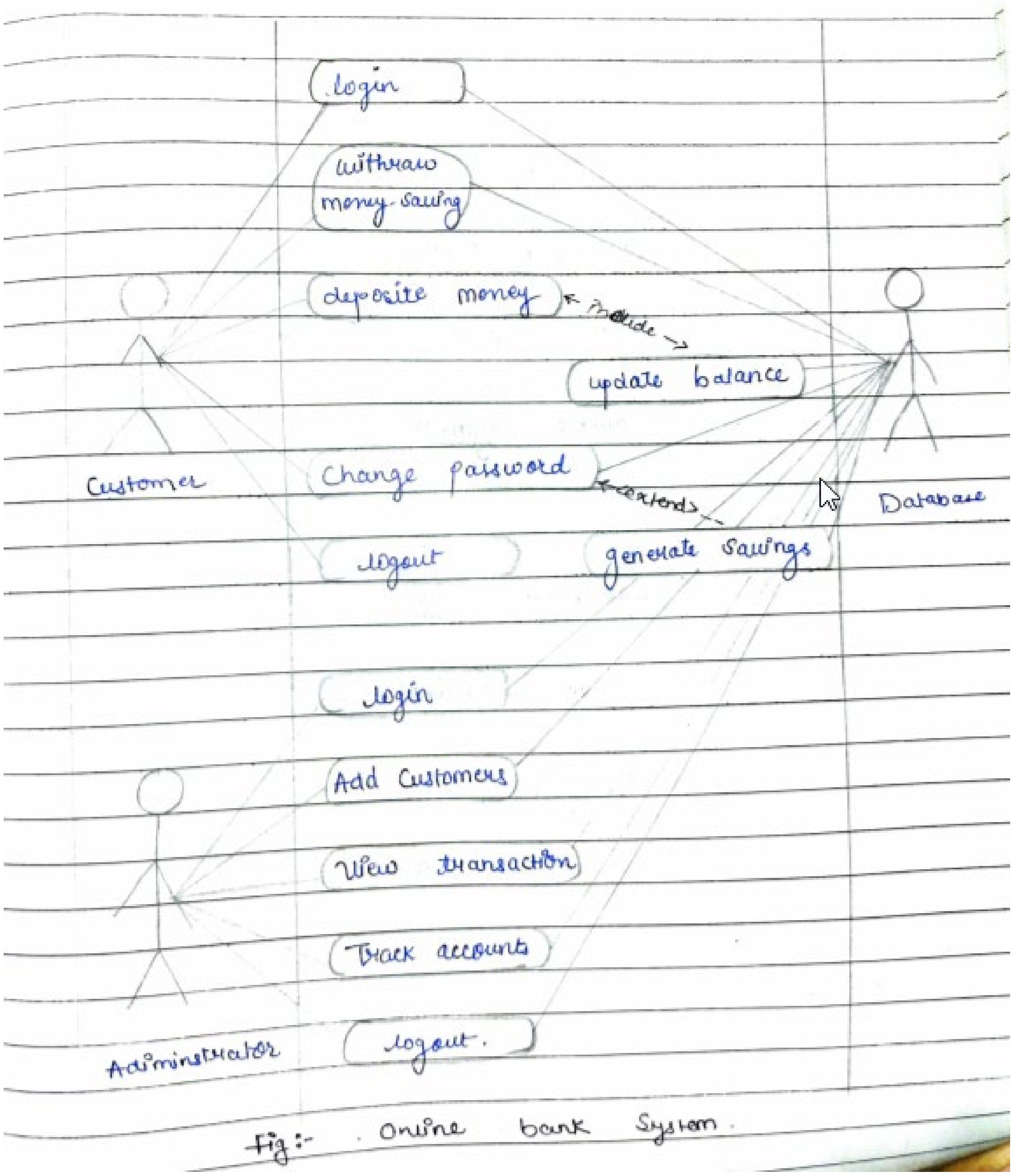


Fig :- Online bank System.

Are use cases the same as functional requirements are different from use cases?

- The use cases describe typical interaction between the users of the system and the system itself. They provide a narrative of how the system is use and each use case.
- The functional requirements describe the systems functionality like
  - o The system should grant access to the user after he provides username and password.
  - o The system should provide access to the following predefined reports, followed by a list of report, etc.
- The difference is in the way the information is presented. The use case describes what should happen whereas the functional requirement may on state the functionality.

Which part of a use case description can also be modelled by using an activity diagram? and why?

In use case description "flow of activities" can be also modelled by using an activity diagram because the flow of the events of use case describes what needs to be done by the system to provide value of an actor. It consists of a sequence of an activities that together produce something for the actor. The flow of event consists of a basic flow, and one or several alternative flows. An activity diagram in the user case model illustration the flow of events of a use case.

Conclusion :

Thus, in this practical I learnt about the use cases diagram, and also made a use case diagram for online banking system.

(10)	(20)	(10)	(10)	TOTAL

EXPERIMENT NO : 06

Aim : Create Sequence diagram, state diagram for given scenario.

Theory:

What is a Sequence diagram?

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 sometimes called event diagrams or event scenarios. Sequence diagrams are preferred by both developers and readers alike for their simplicity.

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.

High-Level Sequence Diagrams:

High-level sequence diagrams give a good overview of the interactions between customers, partners, and the business system. They serve as the basis for the electronic data transfer between the business system and customers, business partners, and suppliers (see Modeling for System Integration).

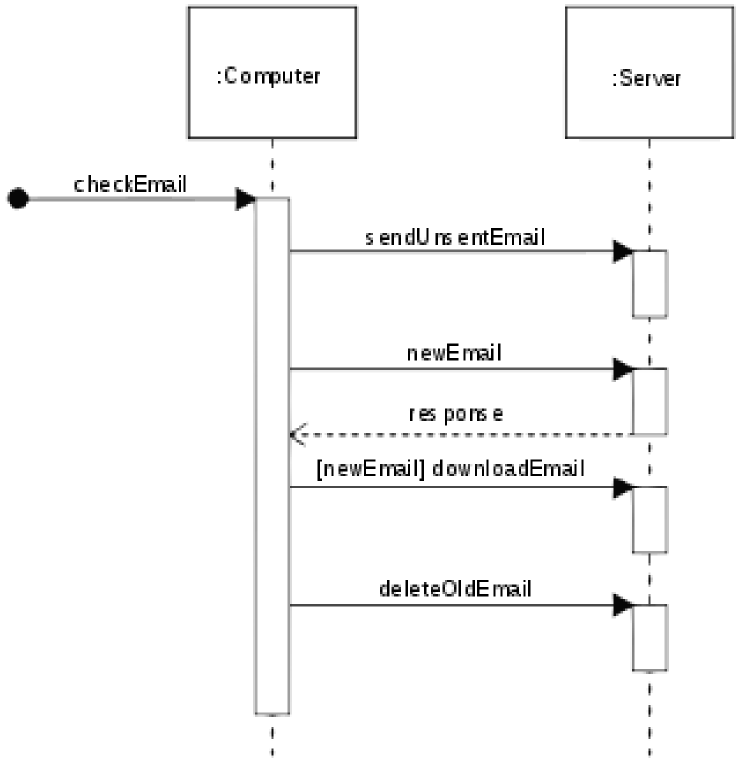


Fig: Example of sequence diagram

What is State diagram?

A state diagram is a type of diagram used in computer science and related fields to describe the behaviour of systems. State diagrams require that the system described is composed of a finite number of states; sometimes, this is indeed the case, while at other times this is a reasonable abstraction. Many forms of state diagrams exist, which differ slightly and have different semantics.

State diagrams are used to give an abstract description of the behavior of a system. This behavior is analyzed and represented as a series of events that can occur in one or more possible states. Hereby "each diagram usually represents objects of a single class and track the different states of its objects through the system".

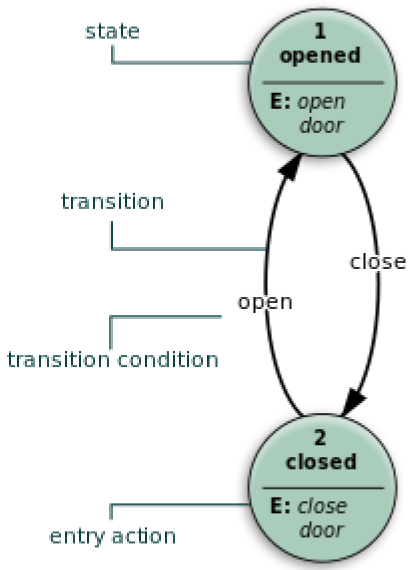


Fig: Example of State diagram

Purpose of state diagram:

Its specific purpose is to define the state changes triggered by events. Events are internal or external factors influencing the system. State chart diagrams are used to model the states and also the events operating on the system.

Questions:

<u>SCENARIO</u> :

1. Government wish to transfer subsidy to those farmer's bank account having land less than 2 hectares.

a. Farmer needs to register themselves on a provided website.

b. Farmer needs to upload required documents.

c. Inspection team inspects farmer document and land. 14

d. Government successfully transfers the subsidy in farmer's bank account.

1. Draw Sequence diagram for given scenario.

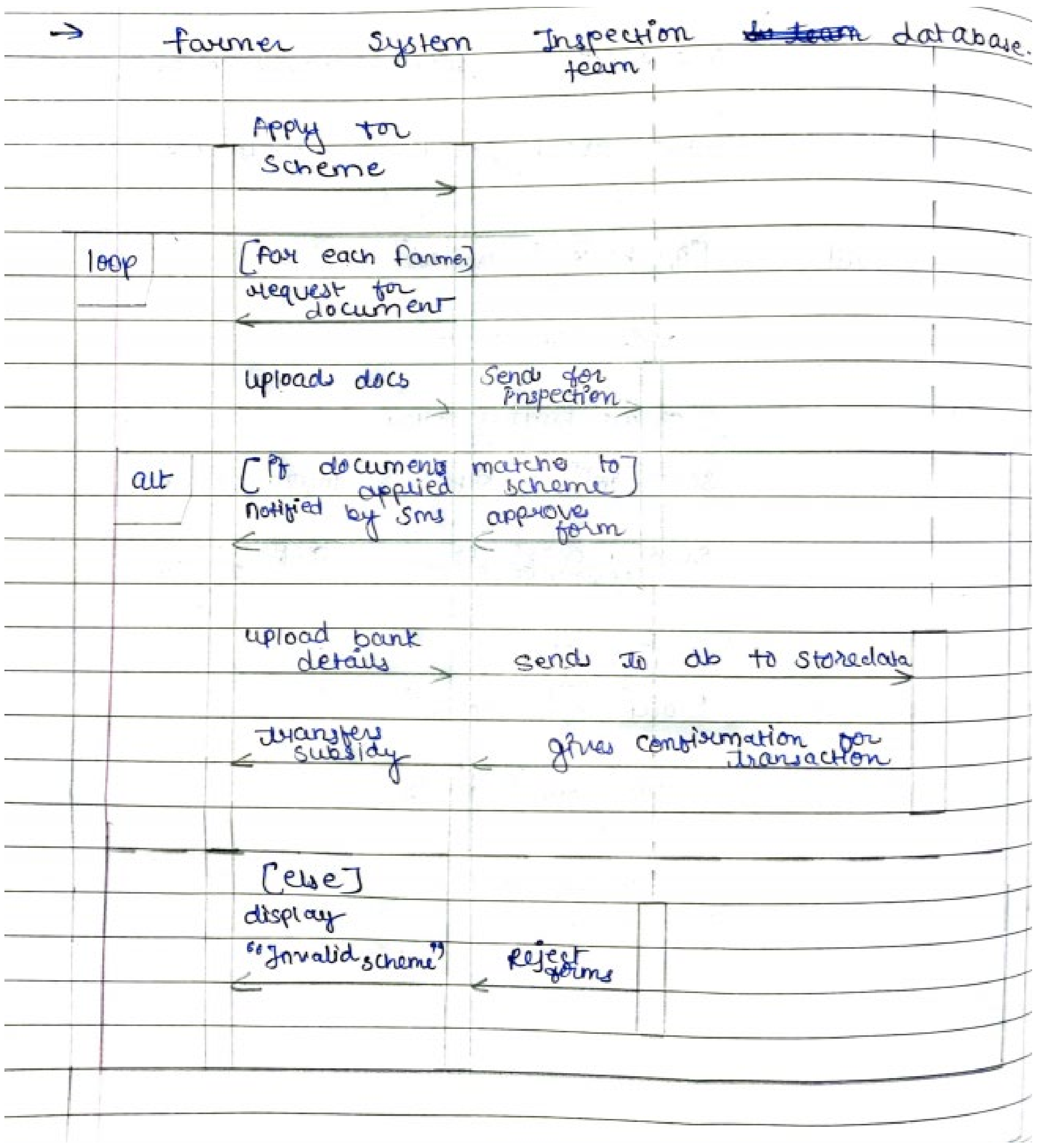


Fig: Sequence diagram for farmer subsidy

2. Draw State diagram for given scenario.

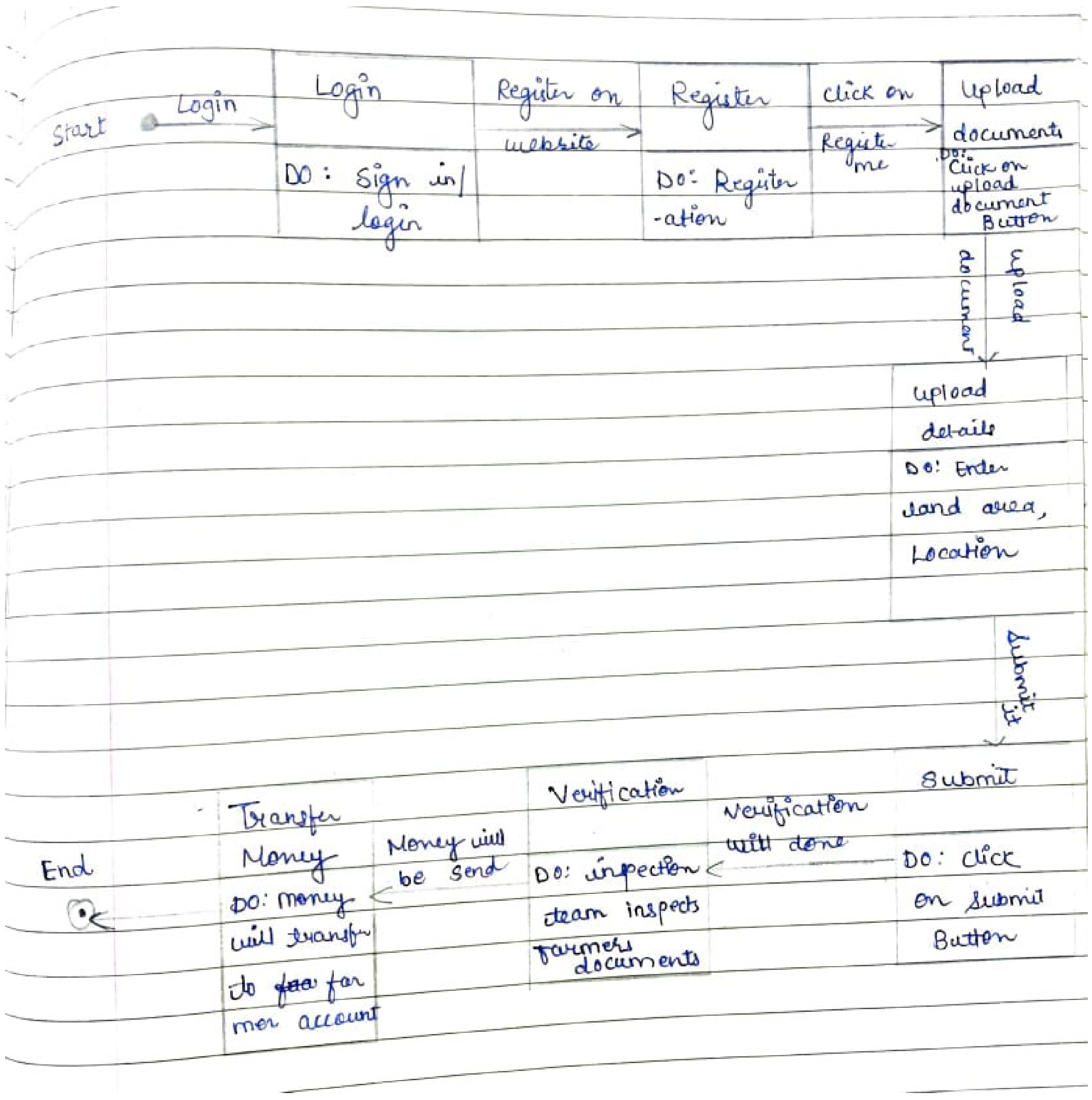


Fig: State diagram for farmer subsidy

3. Comparison between sequence diagram and state diagram.

Sequence diagram	State diagram
1. In sequence diagram the sequence in which an event happen is shown.	1. In state diagram the transition of state is shown.
2. Sequence diagrams are good at showing sequential logic but not that good at gibing “big picture view”.	2. State diagram shows the object undergoing a process. It gives a clear picture of the changes in the objects state in this process.
3. sequence diagram represents behaviour by describing how classes move from state to state.	3. state diagram represents behaviour without nothing the classes involved.

Conclusion:

Thus, in this practical I learnt about state diagram and sequence diagram and also drawn a state and sequence diagram for Government to transfer subsidy to those farmer's bank account and also learnt their comparison.

(10)	(20)	(10)	(10)	TOTAL

EXPERIMENT NO:07

Aim: Draw E-R diagram, DFD and create data dictionary for above system.

Theory:

What is Entity?

An entity is any object in the system that we want to model and store information about. Entities are usually recognizable concepts, either concrete or abstract, such as person, places, things, or events which have relevance to the database.

Some specific examples of entities are Employee, Student, Lecturer.

What is E-R diagram?

An entity relationship model, also called an entity - relationship (ER) diagram, is a graphical representation of entities and their relationships to each other, typically used in computing in regard to the organization of data within databases or information systems.

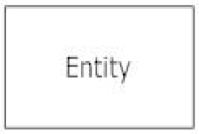
For example: In the following ER diagram we have - two entities Student and College and these two entities have many to one relationship as many student’s study in a single college.

ER Diagram Uses:

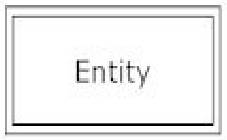
- When documenting a system or process, looking at the system in multiple ways increases the understanding of that system.
- ERD diagrams are commonly used in conjunction with a data flow diaram to display the contents of a data store.
- They help us to visualize how data is connected in a general way and are particularly useful for constructing a relational database.

Common Entity Relationship Diagram Symbols:

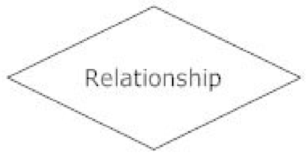
- **Entities**, which are represented by rectangles. An entity is an object or concept about which you want to store information.



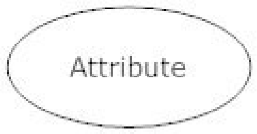
- **weak entity** is an entity that must defined by a foreign key relationship with another entity as it cannot be uniquely identified by its own attributes alone.



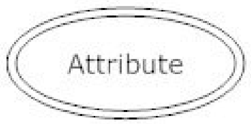
- **Relationship**, which are represented by diamond shapes, show how two entities share information in the database.



- **Attributes**, which are represented by ovals. A key attribute is the unique, distinguishing characteristic of the entity.



- **multivalued attribute** can have more than one value. For example, an employee entity can have multiple skill values.



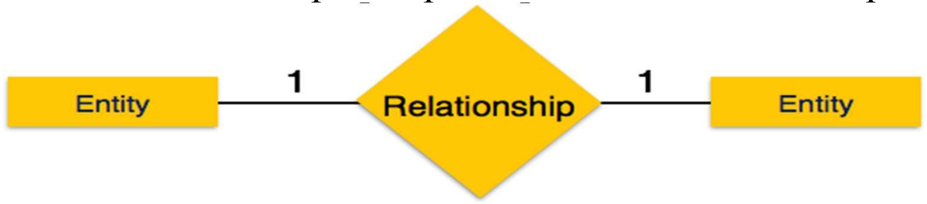
- **Connecting lines**, solid lines that connect attributes to show

the relationships of entities in the diagram.

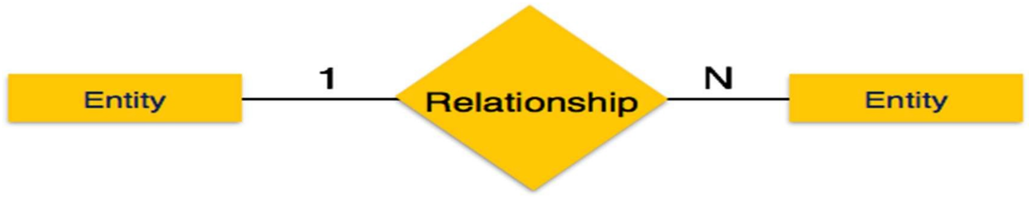
Relationship

Relationships are represented by diamond-shaped box. Name of the relationship is written inside the diamond-box. All the entities (rectangles) participating in a relationship, are connected to it by a line.

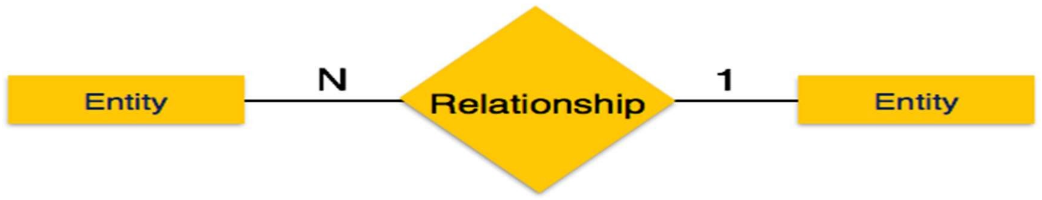
- **One-to-one** − When only one instance of an entity is associated with the relationship, it is marked as '1:1'. The following image reflects that only one instance of each entity should be associated with the relationship. It depicts one-to-one relationship.



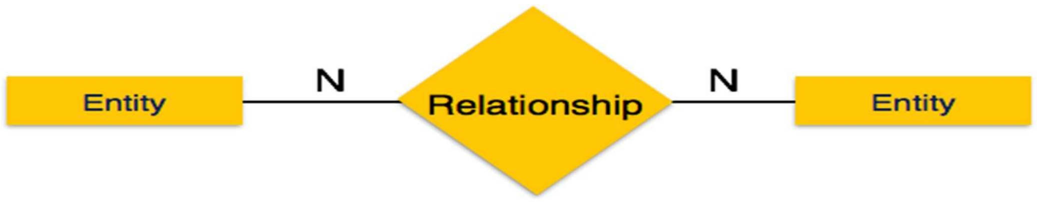
- **One-to-many** − When more than one instance of an entity is associated with a relationship, it is marked as '1:N'. The following image reflects that only one instance of entity on the left and more than one instance of an entity on the right can be associated with the relationship. It depicts one-to-many relationship.



Many-to-one − When more than one instance of entity is associated with the relationship, it is marked as 'N:1'. The following image reflects that more than one instance of an entity on the left and only one instance of an entity on the right can be associated with the relationship



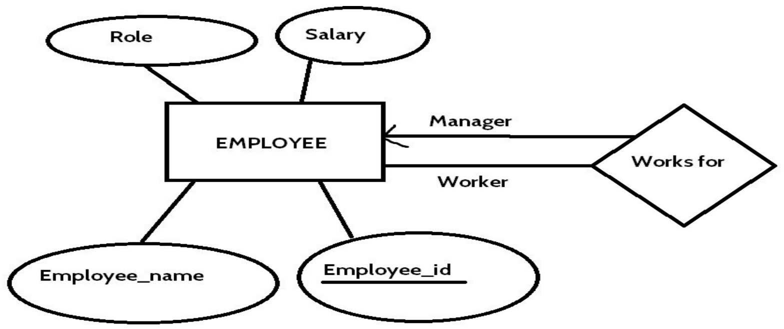
- **Many-to-many** − The following image reflects that more than one instance of an entity on the left and more than one instance of an entity on the right can be associated with the relationship. It depicts many-to-many relationship.



How to Draw ER Diagrams?

Below points show how to go about creating an ER diagram.

1. Identify all the entities in the system. An entity should appear only once in a particular diagram. Create rectangles for all entities and name them properly.
2. Identify relationships between entities. Connect them using a line and add a diamond in the middle describing the relationship.
3. Add attributes for entities. Give meaningful attribute names so they can be understood easily



Advantages of ER Diagram:

- Conceptually it is very simple: ER model is very simple because if we know relationship between entities and attributes, then we can easily draw an ER diagram.
- Better visual representation: ER model is a diagrammatic representation of any logical structure of database. By seeing ER diagram, we can easily understand relationship among entities and relationship.
- Effective communication tool: It is an effective communication tool for database designer.
- Highly integrated with relational model: ER model can be easily converted into relational model by simply converting ER model into tables.
- Easy conversion to any data model: ER model can be easily converted into another data model like hierarchical data model, network data model and so on.

Disadvantages of ER Diagram:

- Limited constraints and specification.
- Loss of information content: Some information be lost or hidden in ER model.
- Limited relationship representation: ER model represents limited relationship as compared to another data models like relational model etc.
- No representation of data manipulation: It is difficult to show data manipulation in ER model.
- Popular for high level design: ER model is very popular for designing high level design

What is Data Flow Diagram?

Data flow diagrams are used to graphically represent the flow of data in a business information system. DFD describes the processes that are involved in a system to transfer data from the input to the file storage and reports generation.

Data flow diagrams can be divided into logical and physical. The logical dataflow diagram describes flow of data through a system to perform certain functionality of a business. The physical data flow diagram describes the implementation of the logical data flow.

DFD Symbols:

There are four basic symbols that are used to represent a data-flow diagram.

Process

A process receives input data and produces output with a different content or form. Processes can be as simple as collecting input data and saving in the database, or it can be complex as producing a report containing monthly sales of all retail stores in the northwest region.

Every process has a name that identifies the function it performs. The name consists of a verb, followed by a singular noun.

Example:

- Apply Payment
- Calculate Commission
- Verify Order

Data Flow

A data-flow is a path for data to move from one part of the information system to another. A data-flow may represent a single data element such the Customer ID or it can represent a set of data element (or a data structure).

Example:

- Customer_info (LastName, FirstName, SS#, Tel #, etc.)
- Order_info (OrderId, Item#, OrderDate, CustomerID, etc.).

Data Store

A data store or data repository is used in a data-flow diagram to represent a situation when the system must retain data because one or more processes need to use the stored data in a later time.

External Entity

It Is also known as actors, sources or sinks, and terminators, external entities produce and consume data that flows between the entity and the system being diagrammed. These data flows are the inputs and outputs of the DFD.

How to draw a data flow diagram?

Lucid chart makes it easy to create a customized data flow diagram starting with a simple template. Choose the symbols you need from our library—processes, data stores, data flow, and external entities— and drag-and-drop them into place. Since Lucid chart is an online tool, it facilitates collaboration and bypasses the hassles of desktop DFD software.

Levels in Data Flow Diagrams (DFD)

In Software engineering DFD (data flow diagram) can be drawn to represent the system of different levels of abstraction. Higher level DFDs are partitioned into low levels-hacking more information and functional elements. Levels in DFD are numbered 0, 1, 2 or beyond. Here, we will see mainly 3 levels in data flow diagram, which are: 0-level DFD, 1-level DFD, and 2-level DFD.

0-level DFD:

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

1-level DFD:

In 1-level DFD, context diagram is decomposed into multiple bubbles/processes.in this level we highlight the main functions of the system and breakdown the high-level process of 0-level DFD into subprocesses.

2-level DFD:

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

Advantages of data flow diagram:

- A simple graphical technique which is easy to understand.
- It helps in defining the boundaries of the system.
- It is useful for communicating current system knowledge to the users.
- It is used as the part of system documentation file.
- It explains the logic behind the data flow within the system.

Disadvantages of data flow diagram:

- Data flow diagram undergoes lot of alteration before going to users, so makes the process little slow.
- Physical consideration are left out. It makes the programmers little confusing towards the system.

What is Data Dictionary?

A data dictionary contains metadata i.e data about the database. The data dictionary is very important as it contains information such as what is in the database, who is allowed to access it, where is the database physically stored etc. The users of the database normally don't interact with the data dictionary, it is only handled by the database administrators.

The different types of data dictionary are:

Active Data Dictionary

If the structure of the database or its specifications change at any point of time, it should be reflected in the data dictionary. This is the responsibility of the database management system in which the data dictionary resides.

Passive Data Dictionary

This is not as useful or easy to handle as an active data dictionary. A passive data dictionary is maintained separately to the database whose contents are stored in the dictionary. That means that if the database is modified the database dictionary is not automatically updated as in the case of Active Data Dictionary.

Creating the Data Dictionary

When you use the Database Configuration Assistant to create a database, Oracle automatically creates the data dictionary. Thereafter, whenever the database is in operation, Oracle updates the data dictionary in response to every DDL statement.

The data dictionary base tables are the first objects created in any Oracle database. They are created in the system tablespace and must remain there. The data dictionary base tables store information about all user-defined objects in the database.

Advantages of data Dictionary:

There are a number of advantages of using Data Dictionary in computer system analysis and design. The advantages are: consistency, clarity; reusability; completeness; increase in sharing and integration; and ease of use for the developer.

- **Scenario**
  1. Library Management System

Library management system is very useful system in school , colleges , In big Library store to maintain records of books , the customers personal information like phone no , address , age , gender , etc. Keeping record of which book is granted to which customer is very necessary .Their due Date , issue date , balance of payment must be recorded in databases for effective and fast working of library . This system will have any effective features like Time-to-Time notification of name of customer that haven't return the books according to their return date , will also show entire status of customer , will keep record of interest of customer and will suggest books according to their interest only.

2. Water Vending Machine

Water Vending Machine will be very useful in public places like railway stations , bus stops , movie halls , public parks , gyms , etc. This can be useful in school , colleges , offices , malls , shops , etc. Water vending machine will allow customer to drink water according to their will and money. This will have features like providing hot water , cold water , warm water , etc . Water will be given in glass, plastic container.

- Draw the ER diagram for given scenario 1?

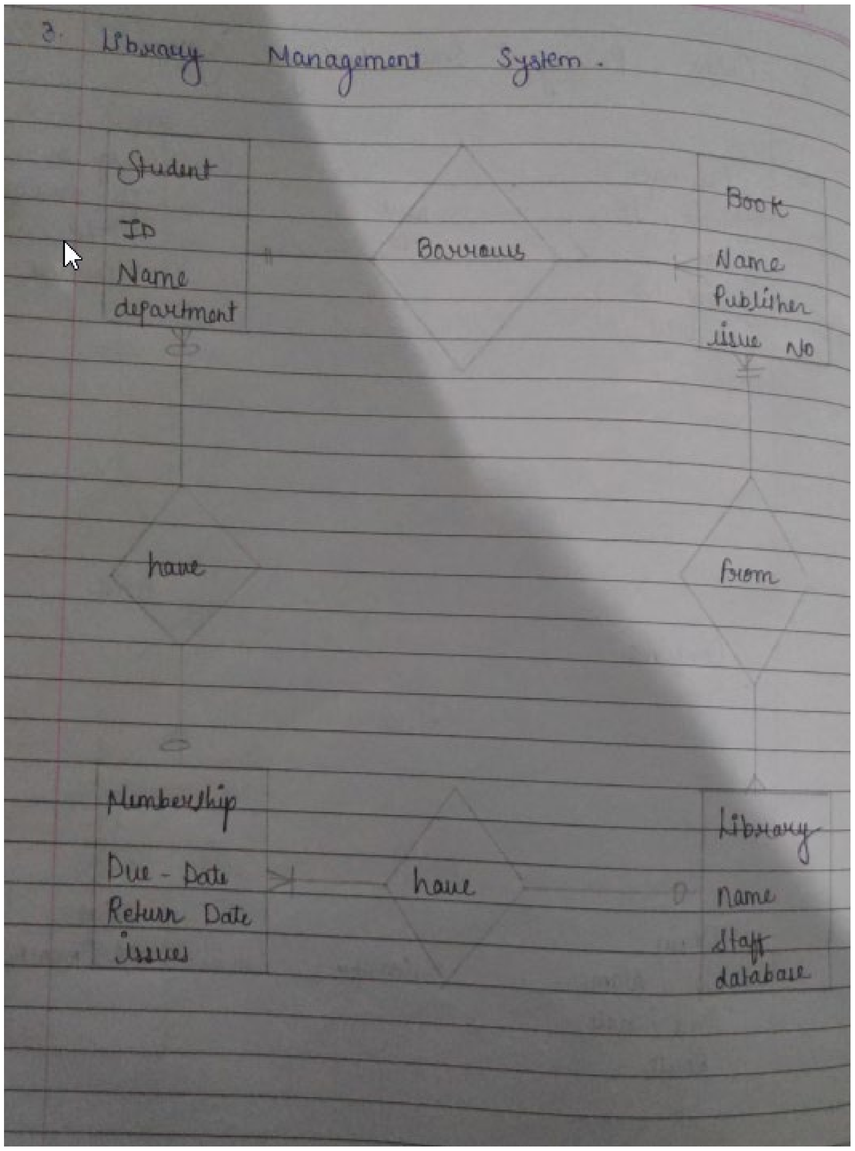


Fig: ER diagram for Library managemnet system

- Draw DFD diagram for given scenario 2?

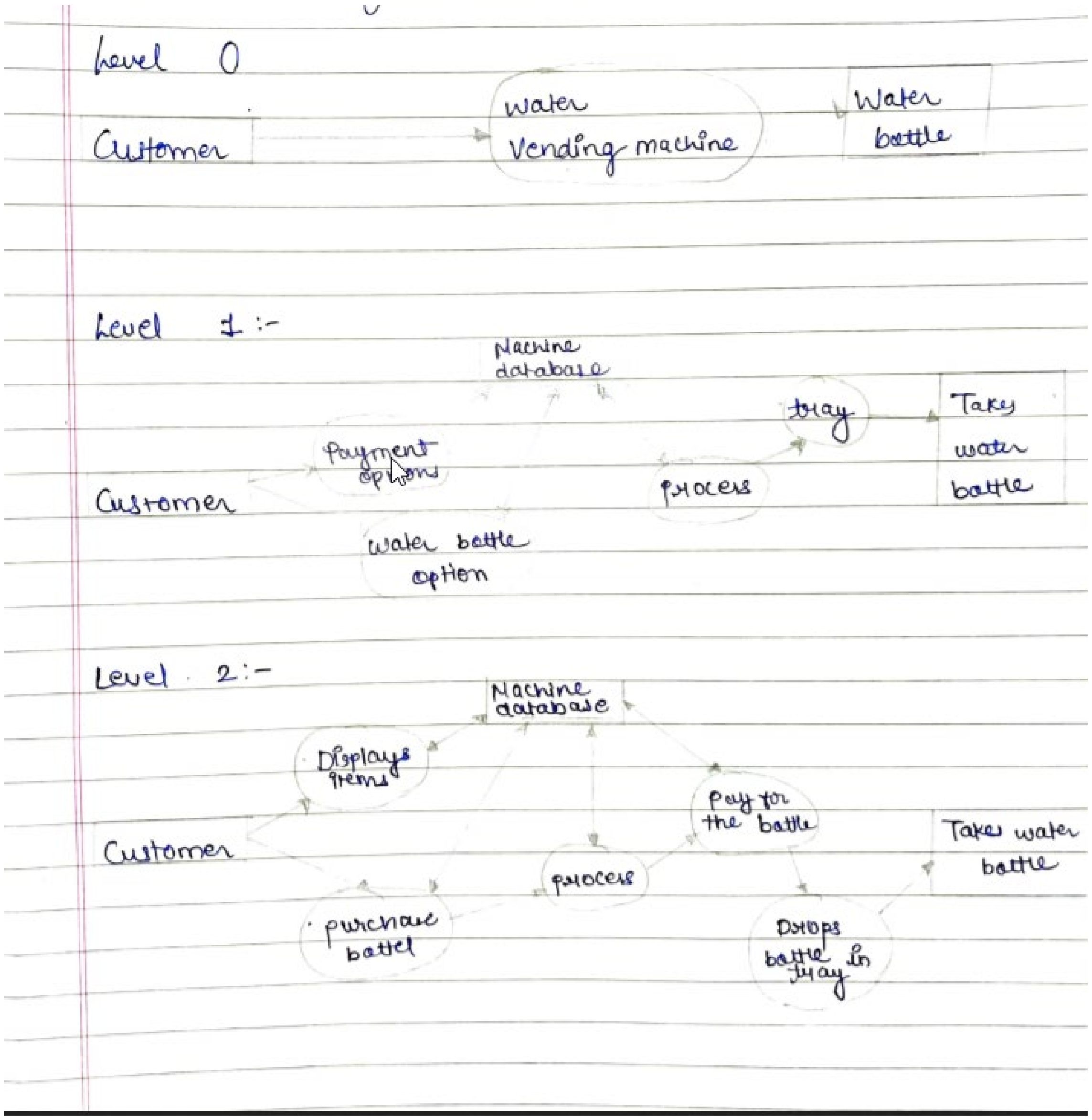


Fig: DFD diagram for water vending machine

- What is the objective of maintaining data dictionary?

The main objective of maintaining data dictionary are:

1. It defines the data objects of each user in the database.
2. When any DDL is fired on the database objects, it searches the data dictionary for the object.
3. It controls the access to different objects in the database by means its view.
4. It provides the quick report on the data and the resources that the objects are using and hence making the management easy.

- CONCLUSION

  Thus, in this practical I have learnt ER diagram, DFD and create data dictionary for the given scenario successfully.

(10)	(20)	(10)	(10)	TOTAL

EXPERIMENT NO. 8

Aim : Draw activity diagram for above system.

Theory :

Activity diagram :

Activity diagram is another important diagram in UML to describe the dynamic aspects of the system**.**

Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system**.**

The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc

The basic purposes of activity diagrams is similar to other four diagrams. It captures the dynamic behavior of the system. Other four diagrams are used to show the message flow from one object to another but activity diagram is used to show message flow from one activity to another.

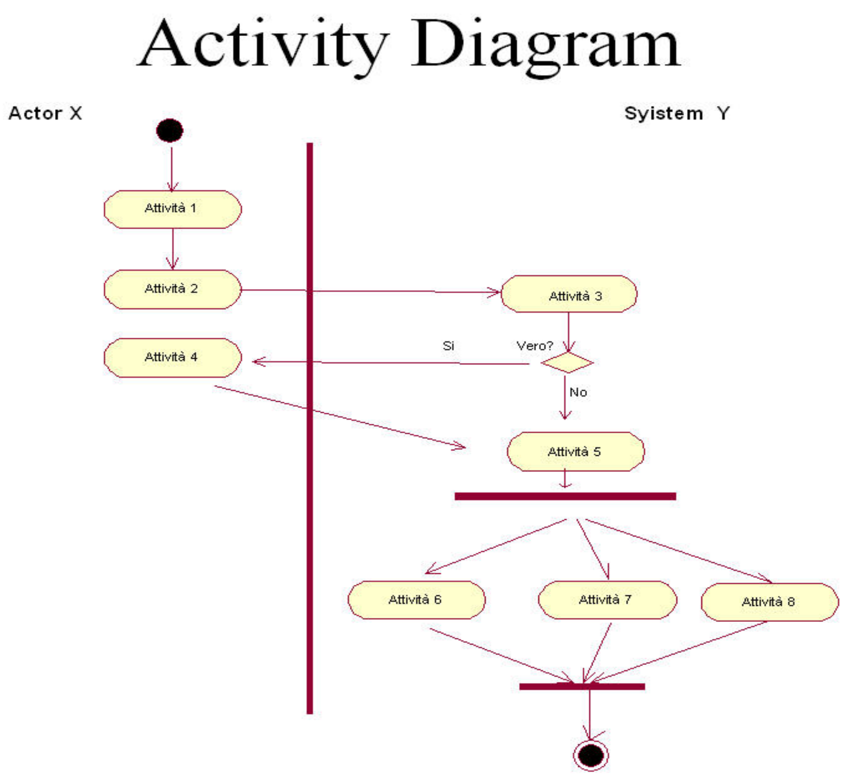


Fig : Activity Diagram

The purpose of an activity diagram can be described as −

- the activity flow of a system.
- Describe the sequence from one activity to another.
- Describe the parallel, branched and concurrent flow of the system.

Activity diagram can be used for −

- Modeling work flow by using activities.
- Modeling business requirements.
- High level understanding of the system's functionalities.
- Investigating business requirements at a later stage.

Scenario: Email account opening and managing system

Questions:

1) When to Use Activity Diagram?

There are some situations like:

1. Understanding workflows: For understanding the activities and the flow.

2. describing a complicated sequential algorithm: The activity diagram is nothing more than a UML.

3. Analysing a use case: We need to understand what action need to take place and what the behavioural dependencies.

4. Dealing with multithreading applications.

2) Draw activity diagram for given scenario.

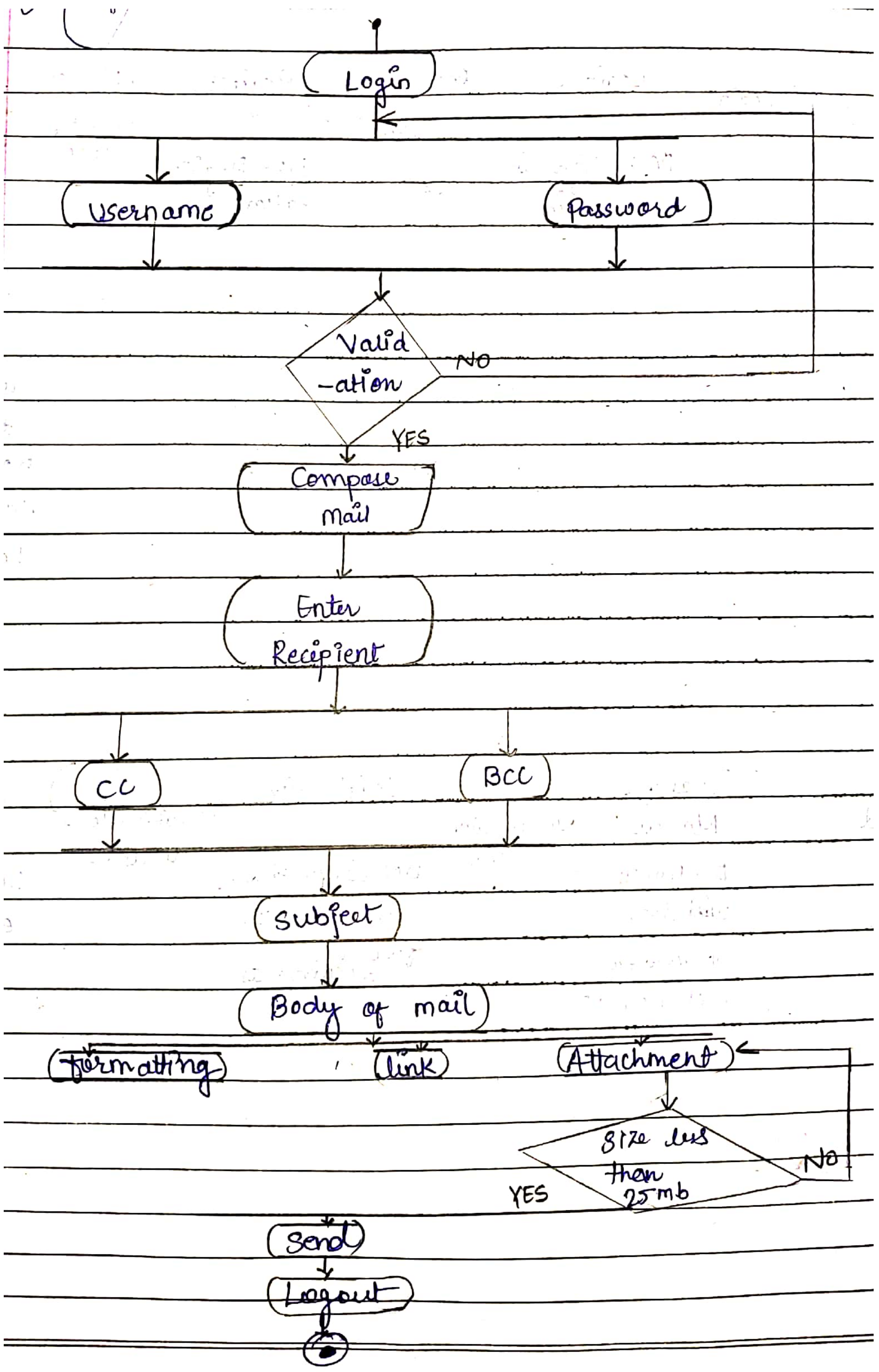


Fig: Activity Diagram for Email account opening

3) Comparison between activity diagram and   sequence diagram.

Sequence diagram	Activity diagram
1. the sequence diagram represent the UML, which is used to visualize sequence of calls in system that is used to perform functionality.	1. the activity diagram represents the UML, which is used to model the workflow of system.
2. sequence diagram is used to represent the time order of a process	2. activity diagram is used to represent the execution of process.
3. The sequence diagram shows the message flow from one object to another object.	3. The activity diagram shows the message flow from one activity to another.
4. The sequence diagram is used for the purpose of dynamic modelling.	4. the activity diagram is used for the purpose of function modelling.

Conclusion :

Thus, in this practical I learnt about activity diagram and also learnt about activity diagram for specific scenario.

10	20	10	10	Total

<u>**EXPERIMENT NO : 09**</u>

Aim : Identify the design principle that is being violated in relation to the given scenario.(Give any Scenario)

Theory :

Design Principles

Software design is both a process and a model. The design process is a sequence of steps that enable the designer to describe all aspects of the software to be built. It is important to note, however, that the design process is not simply a cookbook. Creative skill, past experience, a sense of what makes "good" software, and an overall commitment to quality are critical success factors for a competent design.

There are 10 Design Principles :-

1. **The design process should not suffer from "tunnel vision." :-**

A good designer should consider alternative approaches, judging each based on the requirements of the problem, the resources available to do the job.

2. **The design should be traceable to the analysis model :-**

Because a single element of the design model often traces to multiple requirements, it is necessary to have a means for tracking how requirements have been satisfied by the design model.

3. **The design should not reinvent the wheel :-**

Systems are constructed using a set of design patterns, many of which have likely been encountered before. These patterns should always be chosen as an alternative to reinvention. Time is short and resources are limited! Design time should be invested in representing truly new ideas and integrating those patterns that already exist.

4. **The design should “minimize the intellectual distance” between the software and the problem as it exists in the real world :-**

That is, the structure of the software design should (whenever possible) mimic the structure of the problem domain.

5. **The design should exhibit uniformity and integration :-**

A design is uniform if it appears that one person developed the entire thing. Rules of style and format should be defined for a design team before design work begins. A design is integrated if care is taken in defining interfaces between design components.

6. **The design should be structured to accommodate change :-**

The design concepts discussed in the next section enable a design to achieve this principle.

7. **The design should be structured to degrade gently, even when aberrant data, events, or operating conditions are encountered :-**

Welldesigned software should never “bomb.” It should be designed to accommodate unusual circumstances, and if it must terminate processing, do so in a graceful manner.

8. **Design is not coding, coding is not design :-**

Even when detailed procedural designs are created for program components, the level of abstraction of the design model is higher than source code. The only design decisions made at the coding level address the small implementation details that enable the procedural design to be coded.

9. **The design should be assessed for quality as it is being created not after the fact :-**

A variety of design concepts and design measures are available to assist the designer in assessing quality.

10.The design should be reviewed to minimize conceptual (semantic) errors :-

There is sometimes a tendency to focus on minutiae when the design is reviewed, missing the forest for the trees. A design team should ensure that major conceptual elements of the design (omissions, ambiguity, inconsistency) have been addressed before worrying about the syntax of the design model.

When these design principles are properly applied, the software engineer creates a design that exhibits both external and internal quality factors. External quality factors are those properties of the software that can be readily observed by users (e.g., speed, reliability, correctness, usability). Internal quality factors are of importance to software engineers. They lead to a high-quality design from the technical perspective. To achieve internal quality factors, the designer must understand basic design concepts.

Questions:

SCENARIO : Home security system with camera and sensors

For the safety purpose home security system with cameras and sensors are the best system. This system may reduce the security issue at the house. Where as the CCTV cameras and touch sensors are present to capture the unrequired movements and for hearing voice certain more voice sensors are also attached to give voice. This system can be useful anywhere like school, colleges ,offices, homes, hotels, labs, government buildings, hospitals, etc.

1. Which Principles are being violated with the given scenario.

The home security system with camera and sensors I think design is not coding, coding is not design principle are being violated.

2. How it is violated with the given scenario.

Home security system is system where we need to know about the quality of system devices or system performance. In case of above principle coding is needed and comparison of coding with its design is must while the home security system only requires the device with better quality. In the other words system violated the design is not coding, coding is not design principle because in the system the designer only need to arrange that system as externally but they don"t have an idea about the internal process or coding. This principle is totally different for given scenario.

3. Design is not coding , Coding is not design relate it with the scenario.

Design and coding are different terms. On the basis of given scenario at the time of designing of home security system you need to know only about what type of devices including in system, what they perform and how, what functionalities are required while at the time of coding you need to develop the code on which the system will work and by performing them get the proper results.

Conclusion : Thus, I learnt about the design principle which is necessary to increase the productivity and value of system or software.

(10)	**(20)**	**(10)**	**(10)**	**TOTAL**

Project Report

Submitted by:-

Team Leader:-	Team Members:-
Dipanshu - 1807049	Shamal - 1807004
	Akanksha - 1807037
	Saif - 1807045
	Neha - 1607059

Assignment for Software Engineering

Guided By:-

Lokesh Bhadekar

(Lecturer at Government Polytechnic, Sadar, Nagpur)

Abstract

The main objective of the project is to create an online book store that allows users to search and purchase a book online based on

title, author and subject. The selected books are displayed in a tabular format and the user can order their books online through credit card payment. Post pandemic each person tries to keep himself/herself away from the other so “online book store” allows you to get a contactless home delivery so the user can be safe. Online Book store is an online web application where the customer can purchase books online. Through a web browser the customers can search for a book by its title or author, later can add to the shopping cart and finally purchase using credit card transaction. The user can login using his account details or new customers can set up an account very quickly. They should give the details of their name, contact number and shipping address. The user can also give feedback to a book by giving ratings on a score of five. The books are divided into many categories based on subject like Software, Database, English, and Architecture etc. The Online Book Store Website provides customers with online shopping through a web browser. A customer can, create, sign in to his account, place items into a shopping cart and purchase using his credit card details. The Administrator will have additional functionalities when compared to the common user. He can add, delete and update the book details, book categories, member information and also confirm a placed order. This application is developed using TailwindCSS programming language. The Master page, data sets, data grids, user controls are used to develop the Online Book store.

Table of Contents

1. Introduction

i. Purpose
ii. Scope
iii. Intended audience
iv. Diagrams
v. Reference

Overall Description

i. Product perspective
ii. Product functions
iii. User characteristics
iv. Constraints
v. Assumption and Dependencies

System feature

i. Website Feature
ii. Hardware used
iii. Software Used

External Interface Requirements

i. User Interfaces
ii. Hardware Interface
iii. Software Interface
iv. Communication Interface

What would the website look like?

i. Homepage
ii. Login page
iii. Cart
iv. Bestseller
v. A book with its description

1. Introduction

Purpose

21st Century where every single service is brought online and loved by people. “Online Bookstore” brings book store online. Here the website provides you different services to read/order book. The programing for the website was done in HTML/CSS for front-end and My SQL as database at the back-end. The main purpose of this website is to attract people to read different Genre like Adventure, Mystery, Education, etc. This website provides details of the best seller books on the website.

Scope

Post Covid-19 thinking of people has changed drastically no one wish to go out of the house to get things but get home delivery. This website would help people to do the same it will let its user to preview and order the book and pay online to avoid contact.

The “Online Book Store” is a Web-based Web application that allows User or Costumer to buy a Book as well as the store puts the recommendation of some great achievers and readers which make them to choices among different genre and authors. The application should be free to use from either a mobile Brower or similar services.

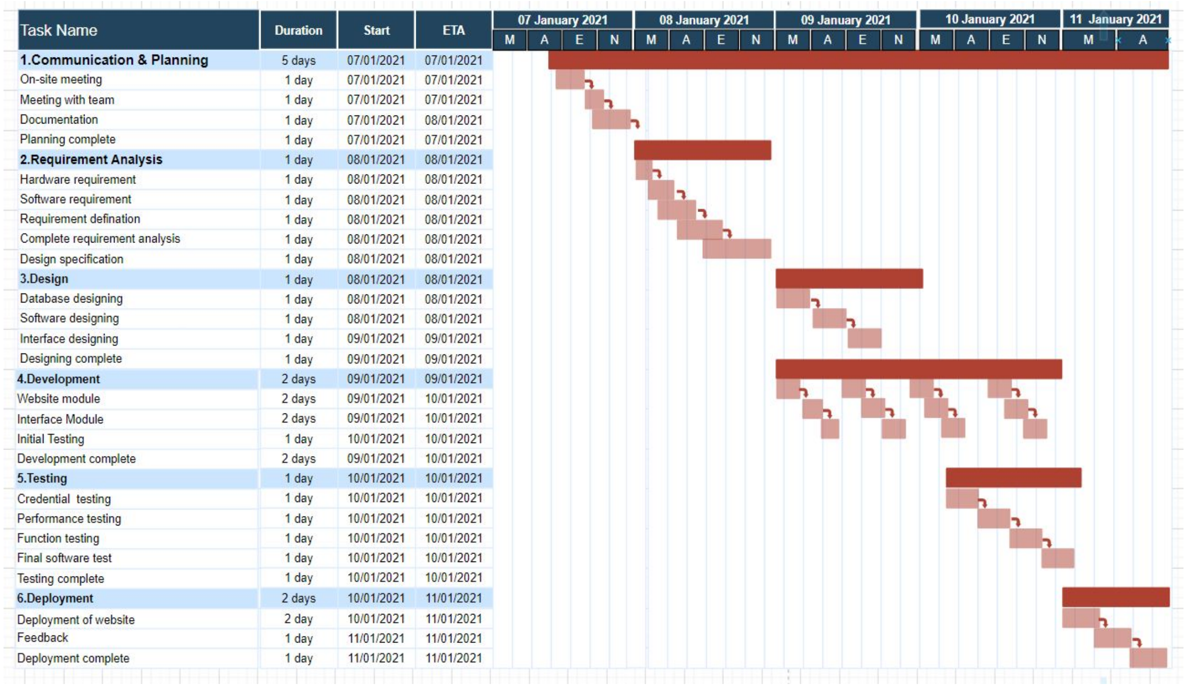
Furthermore, the Web needs Internet and to fetch and display results. All system information is maintained in a database, which is located on a web-server. The application also has the capability of representing both summary and detailed information about a book.

Intended audience

The intended audience for website would be people of all age we love to read and explore things. The site attract user with its unique feature of being able to buy book in different formats. It not only allows you to buy hard copy but also audio format and kindle format also.

Reference

1) https://www.daitm.org.in/wp-content/uploads/2019/04/15499016006_SubarnaDay.pdf
2) https://www.sites.google.com/site/bcafinalyearproject/project-report/online-book-store-project-report
3) https://lecturenotes.in/project-report/21350-online-book-shop-project-report/5



Task Name	Duration	Start	ETA
1.Communication & Planning	5 days	07/01/2021	07/01/2021
On-site meeting	1 day	07/01/2021	07/01/2021
Meeting with team	1 day	07/01/2021	07/01/2021
Documentation	1 day	07/01/2021	08/01/2021
Planning complete	1 day	07/01/2021	07/01/2021
2.Requirement Analysis	1 day	08/01/2021	08/01/2021
Hardware requirement	1 day	08/01/2021	08/01/2021
Software requirement	1 day	08/01/2021	08/01/2021
Requirement defination	1 day	08/01/2021	08/01/2021
Complete requirement analysis	1 day	08/01/2021	08/01/2021
Design specification	1 day	08/01/2021	08/01/2021
3.Design	1 day	08/01/2021	08/01/2021
Database designing	1 day	08/01/2021	08/01/2021
Software designing	1 day	08/01/2021	08/01/2021
Interface designing	1 day	09/01/2021	09/01/2021
Designing complete	1 day	09/01/2021	09/01/2021
4.Development	2 days	09/01/2021	09/01/2021
Website module	2 days	09/01/2021	10/01/2021
Interface Module	2 days	09/01/2021	10/01/2021
Initial Testing	1 day	10/01/2021	10/01/2021
Development complete	2 days	09/01/2021	10/01/2021
5.Testing	1 day	10/01/2021	10/01/2021
Credential testing	1 day	10/01/2021	10/01/2021
Performance testing	1 day	10/01/2021	10/01/2021
Function testing	1 day	10/01/2021	10/01/2021
Final software test	1 day	10/01/2021	10/01/2021
Testing complete	1 day	10/01/2021	10/01/2021
6.Deployment	2 days	10/01/2021	11/01/2021
Deployment of website	2 day	10/01/2021	11/01/2021
Feedback	1 day	11/01/2021	11/01/2021
Deployment complete	1 day	11/01/2021	11/01/2021

Diagram

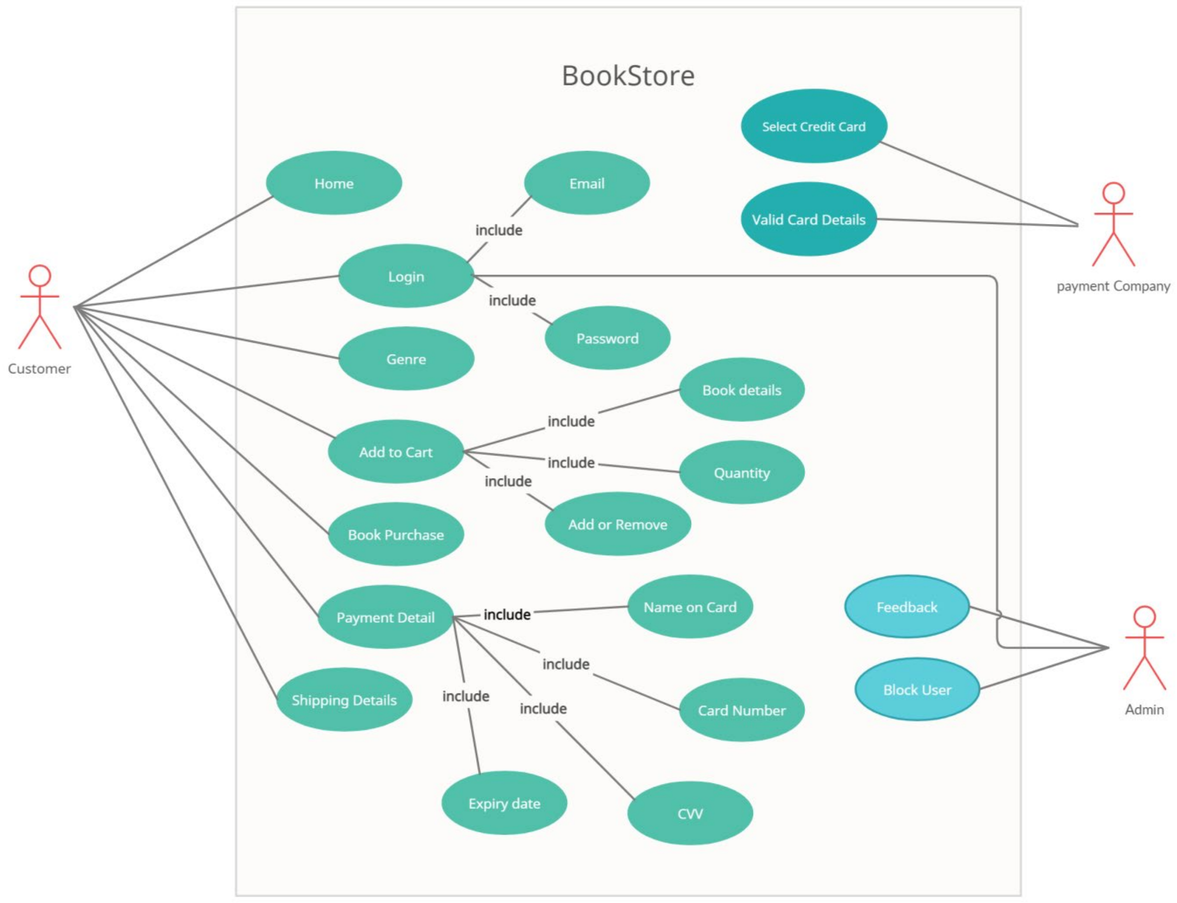


Fig.1 Use-case diagram for Online book store

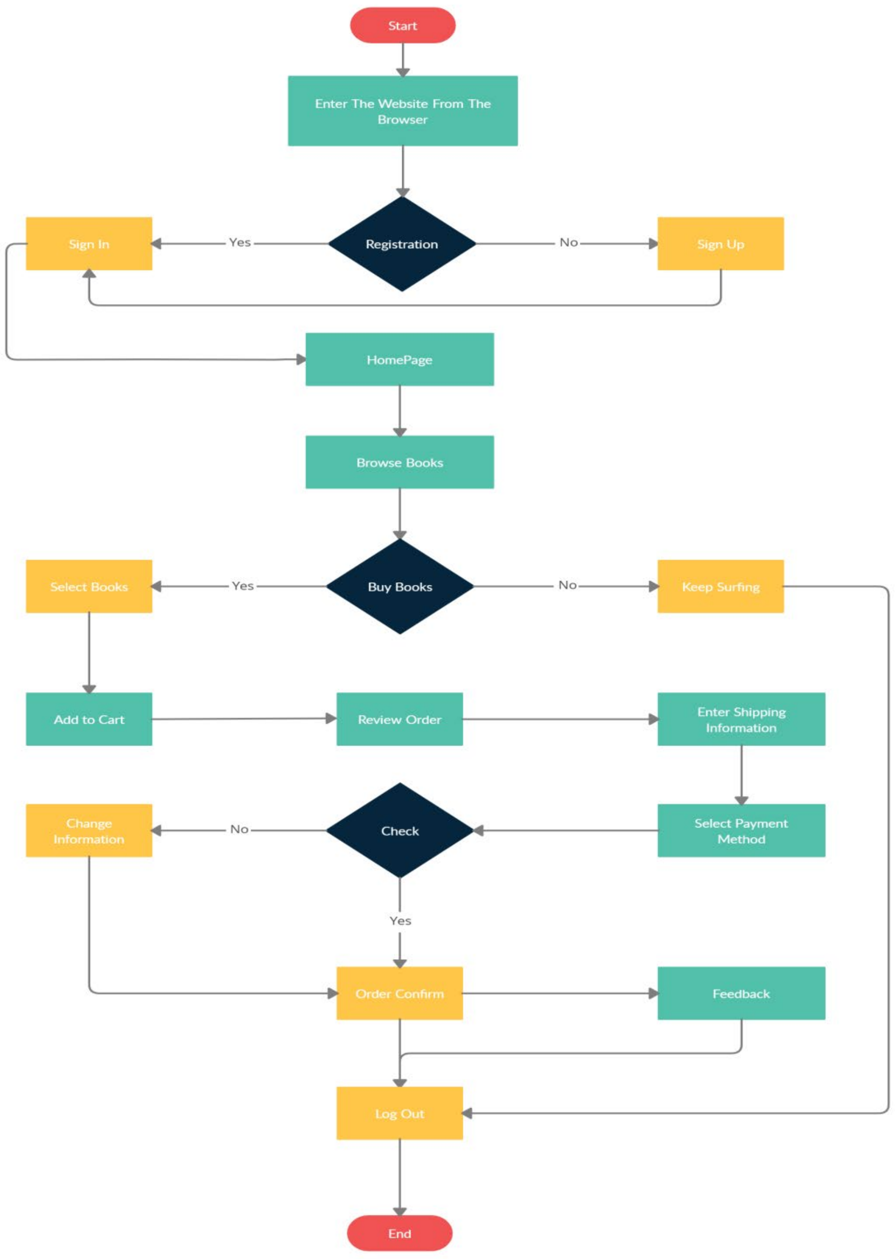


Fig.2 Data flow diagram

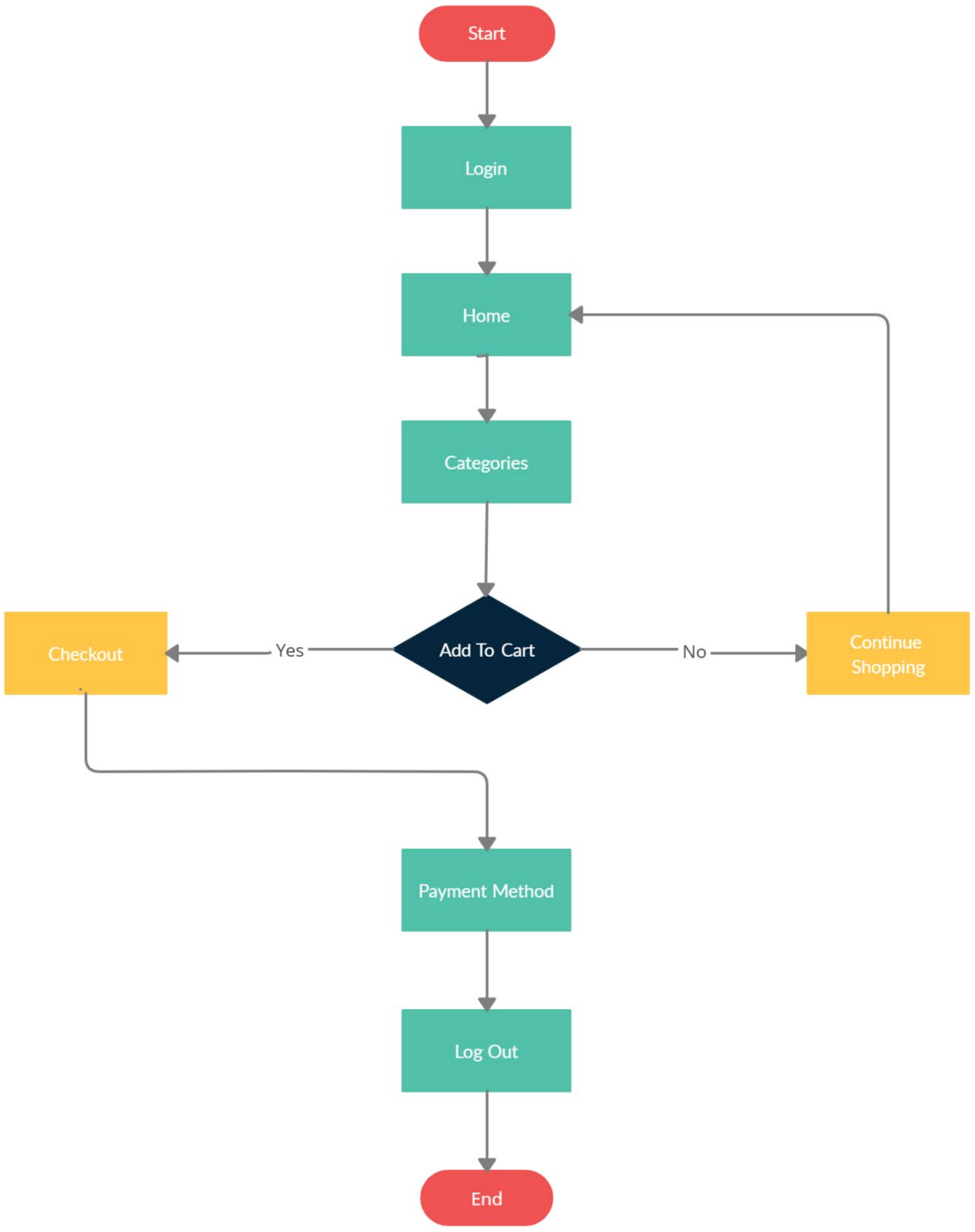


Fig.3 Data-flow diagram for buying book.

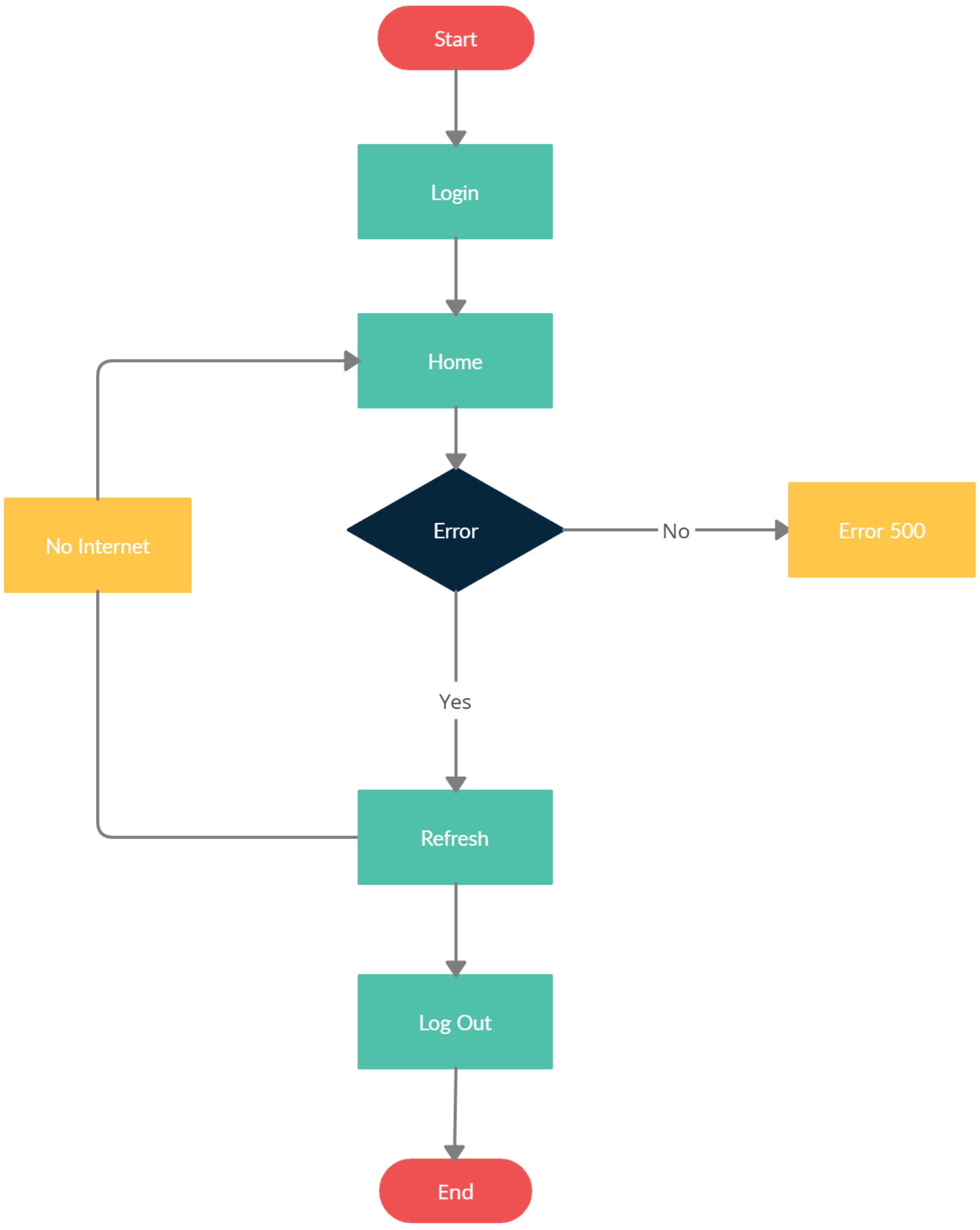


Fig.4 Data-flow diagram for Error

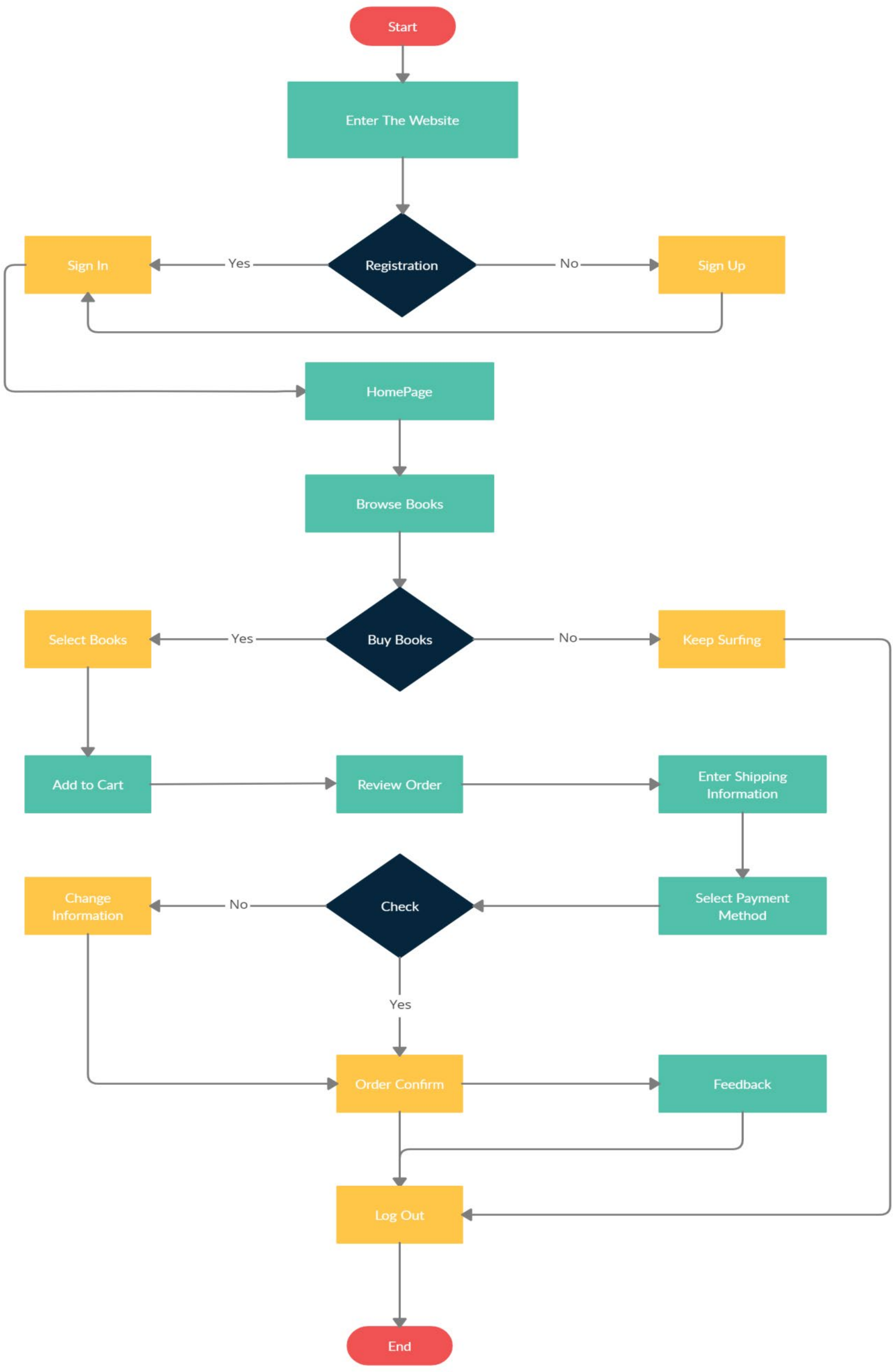


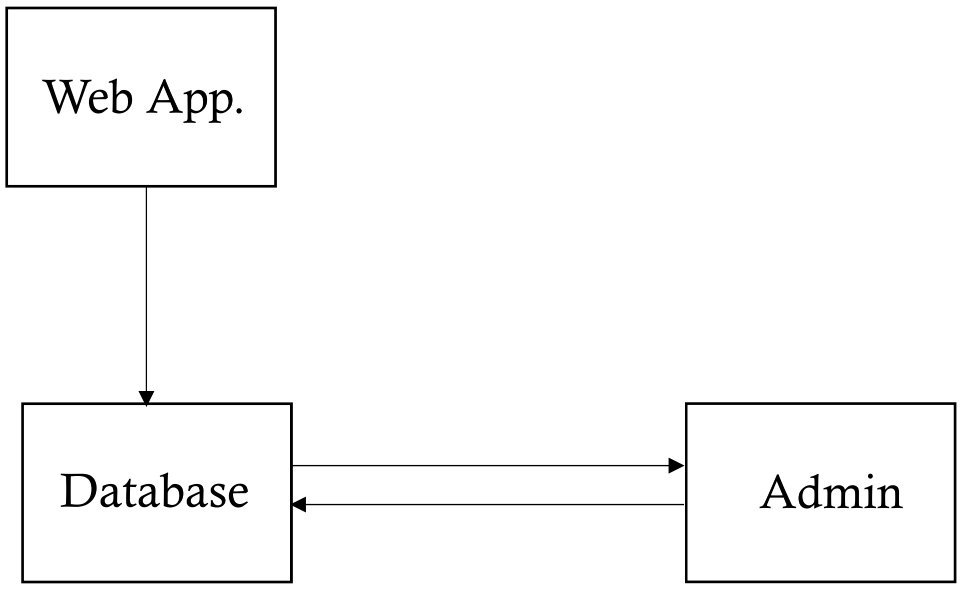
Fig.4 Buying book

2. Overall Description

Product perspective

This system will consist of two parts: one web application and one web portal. The web application will be used to find books and view information about them while the web portal will be used for managing the information about the books and the system as a whole.

The web application will need to communicate with database that is maintained in the server. So the device in which web application is been operated needs to be to internet to communicate with the server.



Product functions

We have implemented and used Waterfall model for the development of the web application.

Login/Sign up: - The very first page that appears in front of the user is to create or log in to the account.

Homepage: -After the user logs into his/her account homepage appears with the treading books on the website.

Genre: - This button on the top bar of the website shows a drop-down list and allows you find the book according to the genre.

Best Seller: - This button leads you the sites best-seller books.

Contact Us: - This feature of the website allows you to send us message about the review, books you would like to see, or send us feedback about the website.

Cart: - This feature of the website will allow user to add books to cart increase or decrease amount of books to order, pay the amount, check the bill, add shipping address.

User characteristics

Login: - The user is allowed to login to the website with the help of user-id and password also in case if the user forgets the password he/she can reset it.

Create account: - The user is allowed to create account by entering a few details like Full name, Email, Password, confirm Password.

Buy book: - This module allows user to enter the books to cart, add payment details, and shipping address for delivery.

A unique feature that is provided to user is that he/she can not only buy a hard copy of the book but also e-book and Audio book the user can carry it anywhere and can read any time.

Constraints

The web application is constrained by the system interface to browser system within the mobile phone. Since there are multiple systems and multiple browsers, the interface will most likely not be the same for every one of them. Also, there may be a difference between what cookies features each of them provide.

The Internet connection is also a constraint for the web application. Since the web application fetches data from the database over the Internet, it is crucial that there is an Internet connection for the application to function.

Both the web portal and the mobile application will be constrained by the capacity of the database. Since the database is shared between both application it may be forced to queue incoming requests and therefor increase the time it takes to fetch data.

Assumption and Dependencies

One assumption about the product is that it will always be used on mobile phones that have enough performance or a nice Browser. If the phone does not have enough hardware resources available for the application, for example the users might have allocated them with other applications; there may be scenarios where the application does not work as intended or even at all.

Another assumption is that the internet components in all devices work in the same way. If the device has different interfaces to the browser, the application need to be specifically adjusted to each interface would mean the integration with the internet would have different requirements than what is stated in this specification.

Another Dependency is that a User must have an account. If a user does not have an account, he/she has to make an account no matter what. Not having an account, then portals will not lead you to nor buy a product neither browse the site.

3. System feature

Website Feature:-

Login/Sign up: - The very first page that appears in front of the user is to create or log in to the account.

Create account: - The user is allowed to create account by entering a few details like Full name, Email, Password, confirm Password.

Homepage: -After the user logs into his/her account homepage appears with the treading books on the website.

Genre: - This button on the top bar of the website shows a drop-down list and allows you find the book according to the genre.

Best Seller: - This button leads you the sites best-seller books.

Contact Us: - This feature of the website allows you to send us message about the review, books you would like to see, or send us feedback about the website.

Cart: - This feature of the website will allow user to add books to cart increase or decrease amount of books to order, pay the amount, check the bill, and add shipping address.

A unique feature that is provided to user is that he/she can not only buy a hard copy of the book but also e-book and Audio book the user can carry it anywhere and can read any time.

Hardware used:-

Asus ROG G531GT

I5 9th gen

Nvidia Geforce GTX 1650 4GB

24 GB RAM

512GB SSD

1TB HDD

15.6 full hd Display

3hr Battery Backup

Software Used: -

Atom: - Atom is a free and open-source text and source code editor for macOS, Linux, and Microsoft Windows with support for plug-ins written in JavaScript, and embedded Git Control, developed by GitHub. Atom is a desktop application built using web technologies. Most of the extending packages have free software licenses and are community-built and maintained. Atom is based on Electron (formerly known as Atom Shell), a framework that enables cross-platform desktop applications using Chromium and Node.js. It is written in CoffeeScript and Less.

Atom was released from beta, as version 1.0, on 25 June 2015. Its developers call it a "hackable text editor for the 21st Century". It is fully customizable in HTML, CSS, and JavaScript.

Chrome: - Google Chrome is a cross-platform web browser developed by Google. It was first released in 2008 for Microsoft Windows, and was later ported to Linux, macOS, iOS, and Android where it is the default browser built into the OS. The browser is also the main component of Chrome OS, where it serves as the platform for web applications.

Most of Chrome's source code comes from Google's free and open-source software project Chromium, but Chrome is licensed as proprietary freeware. WebKit was the original rendering engine, but Google eventually forked it to create the Blink engine; all Chrome variants except iOS now use Blink.[14]

4. External Interface Requirements

This section provides details of all inputs and outputs including hardware, software, communication and mockup prototype.

User Interfaces

Online book store should contain following user interfaces,Login page for authenticating registered users. This screen should accept user id, password and authenticate against corporate authentication system. It also provides features for "New user registration" and "Forgot password" and "Forgot user id".Product search page where registered user can search product based on product attributes. User can search by product name, brief description, and product category and product id. Search should support intuitive features such as type-ahead, synonym support, categorized results grouping and spell correction.

Online book purchase page displays the existing items in the purchase cart along with total amount and allows the user to check out.

Checkout page allows the user to purchase the products using credit/debit card or using PayPal account. It should be integrated with payment gateway and display the order details after successful payment. A confirmation email should be sent to registered email after successful completion.

Hardware Interface

There is no direct hardware interface specifically for online book store. The web application runs on an application server hosted in-house on enterprise hardware.

Software Interface

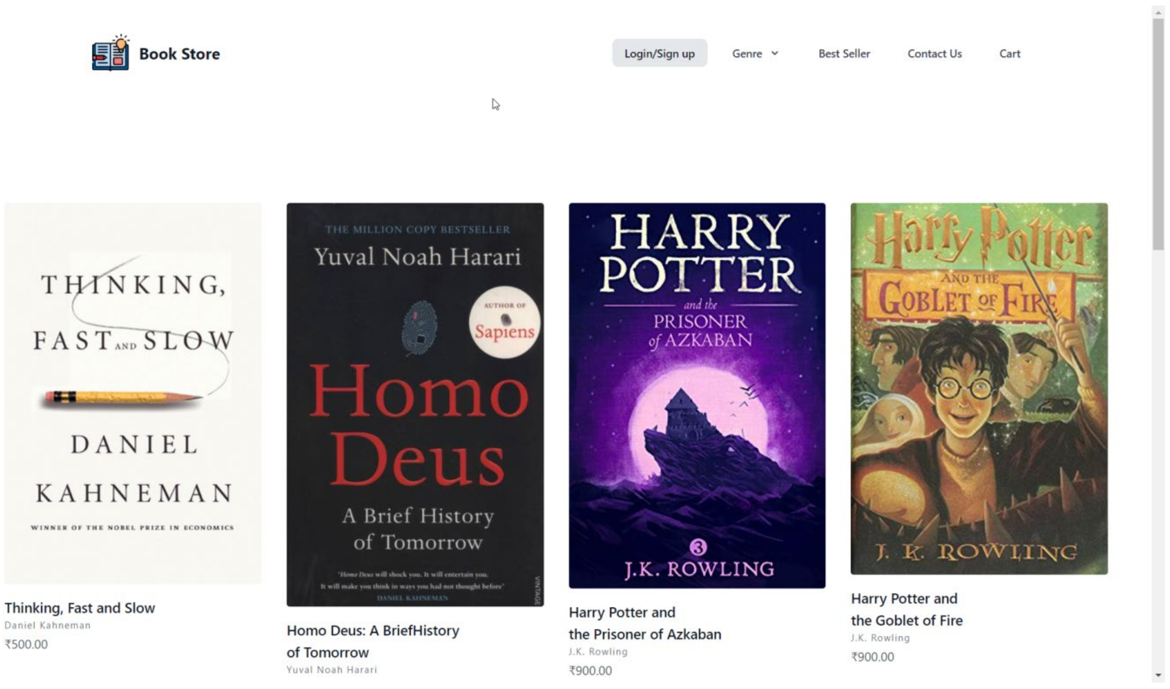
Online book store should integrate with the following interfaces, Product database to get product details. JDBC APIs are the most preferred way of integration. of integration.
Pricing System to get the product pricing, in real-time for the selected products. Integration should be done using web services.
Inventory ERP to get the product availability information. Integration should be done using web services.

Communication Interface

There are no online book specific communication interface requirements. Existing OS and network infrastructure will be leveraged for communication.

5. What would the website look like?

a) Homepage



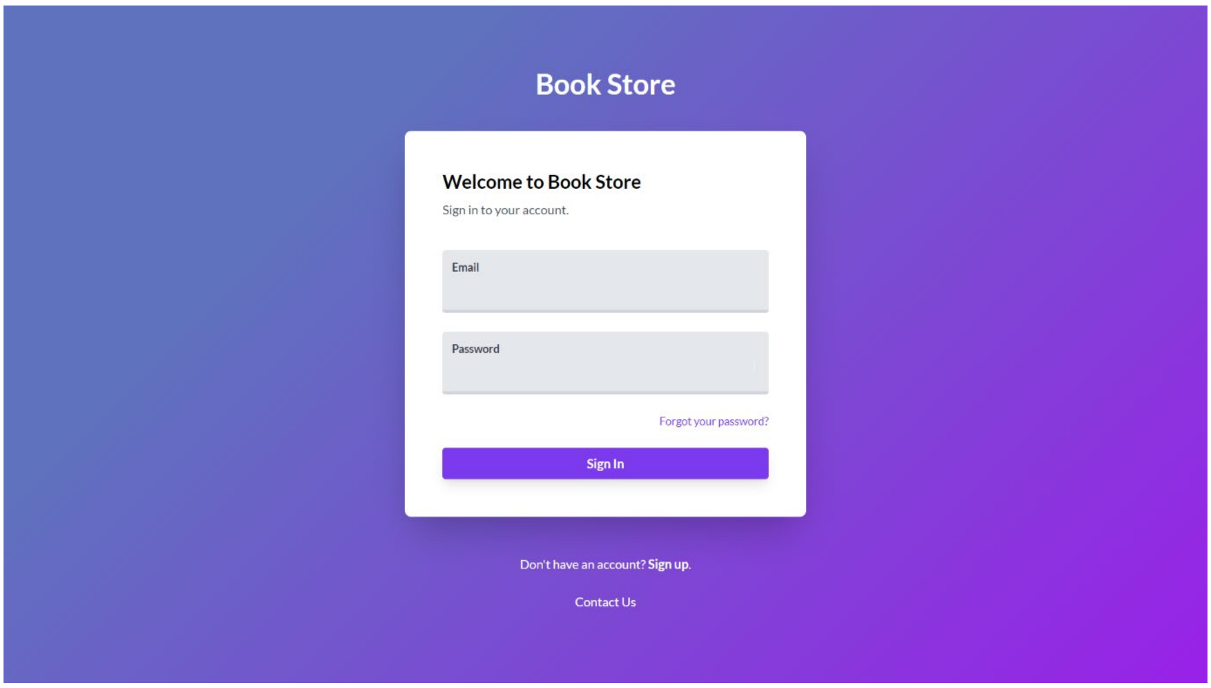
Thinking, Fast and Slow
Daniel Kahneman
₹500.00

Homo Deus: A BriefHistory of Tomorrow
Yuval Noah Harari

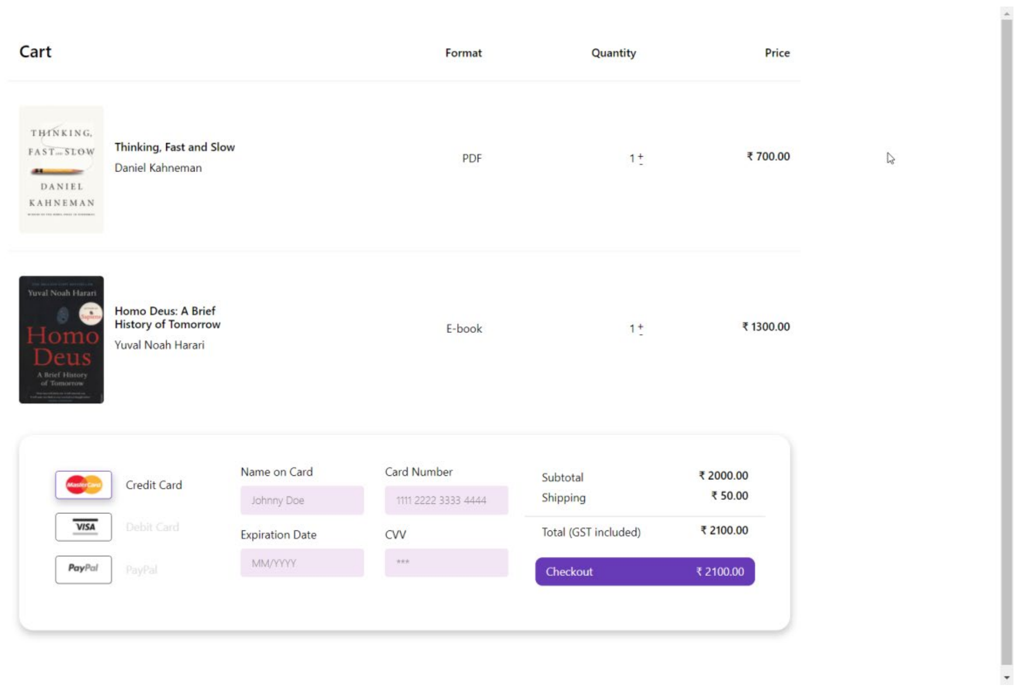
Harry Potter and the Prisoner of Azkaban
J.K. Rowling
₹900.00

Harry Potter and the Goblet of Fire
J.K. Rowling
₹900.00

b) Login page



c) Cart



Cart	Format	Quantity	Price
Thinking, Fast and Slow Daniel Kahneman	PDF	1	₹ 700.00
Homo Deus: A Brief History of Tomorrow Yuval Noah Harari	E-book	1	₹ 1300.00

Credit Card

Debit Card

PayPal

Name on Card: Johnny Doe

Card Number: 1111 2222 3333 4444

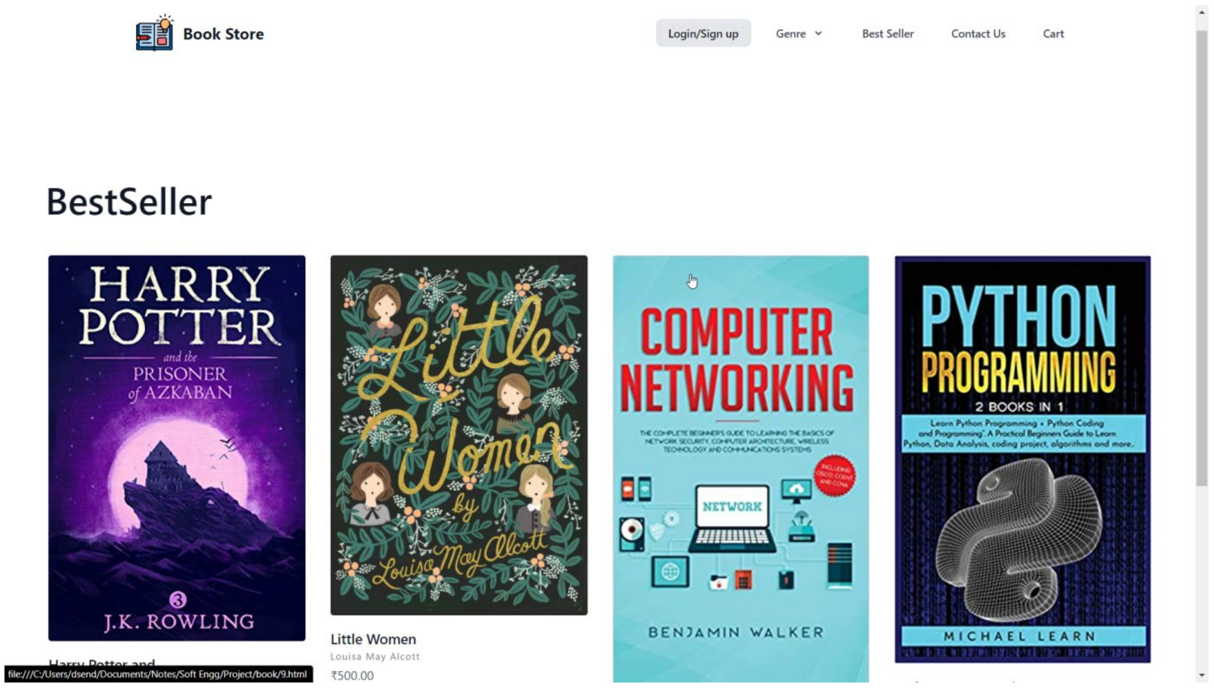
Expiration Date: MM/YYYY

CVV: ***

Subtotal	₹ 2000.00
Shipping	₹ 50.00
Total (GST included)	₹ 2100.00

Checkout ₹ 2100.00

d) Bestseller



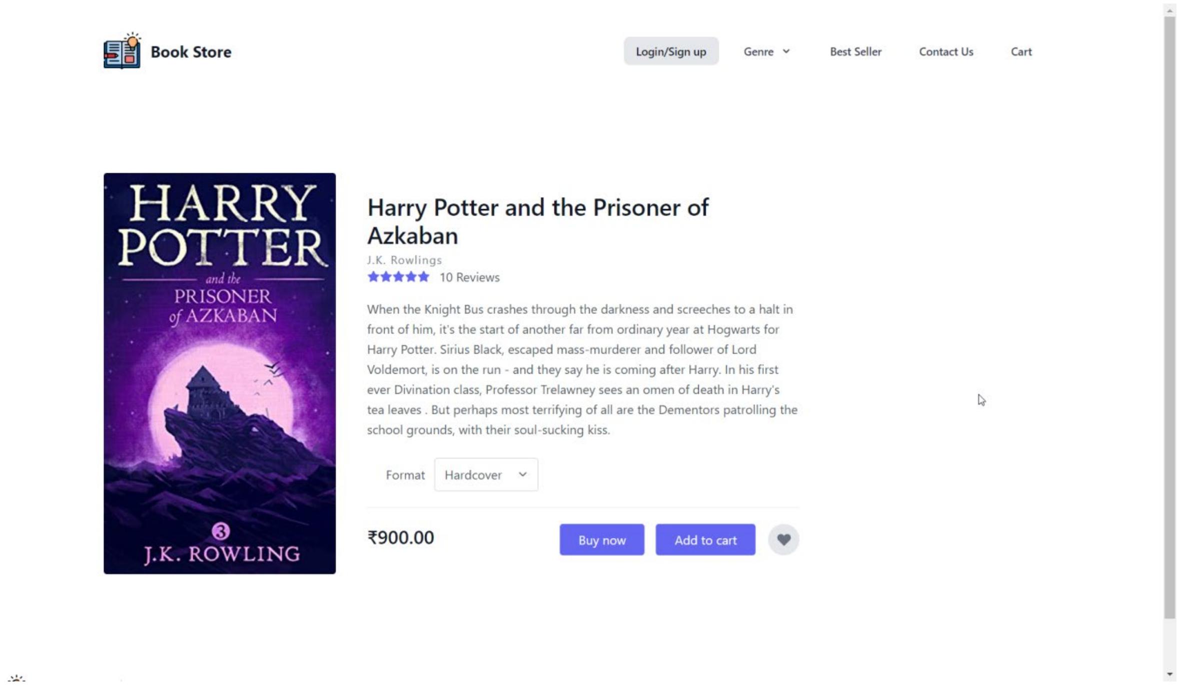
BestSeller

Harry Potter and

Little Women
Louisa May Alcott
₹500.00

file:///C:/Users/dsend/Documents/Notes/Soft Engg/Project/book/9.html

e) A book with its description



Harry Potter and the Prisoner of Azkaban

J.K. Rowlings

10 Reviews

When the Knight Bus crashes through the darkness and screeches to a halt in front of him, it's the start of another far from ordinary year at Hogwarts for Harry Potter. Sirius Black, escaped mass-murderer and follower of Lord Voldemort, is on the run - and they say he is coming after Harry. In his first ever Divination class, Professor Trelawney sees an omen of death in Harry's tea leaves . But perhaps most terrifying of all are the Dementors patrolling the school grounds, with their soul-sucking kiss.

Format: Hardcover

₹900.00

Efforts Made by the Team:-

Dipanshu: - Website coding

Akanksha: - Report

Neha: - Diagram making

Shamal & Saif: - Making SRS document

Conclusion:- Online book store is an online web application where the customer can purchase book online. Through a web browser the customer can search for the book by surfing the website and later can add to the shopping cart and finally purchase the book

End of Report