

## **EXPERIMENT NO: 01**

**Aim:** 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.

1. **Advantages:**

1. It is simple and easy.
2. Easy to manage.

3. It is good for low budget small project.

2. Disadvantages:

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

3. Diagram:

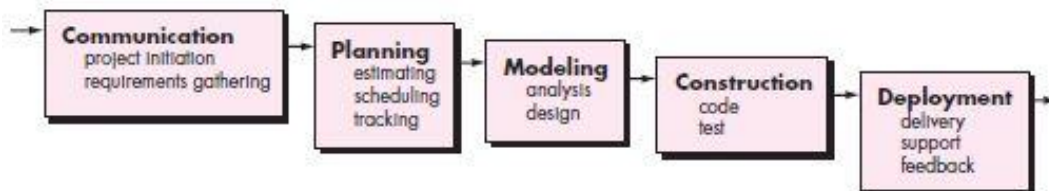


Fig.1.1 Waterfall Model

- **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, constructions 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.

4. 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.

5. 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.
6. Disadvantages:
1. Actual cost will be more than Estimated cost.
  2. Needs good planning and design.
7. Diagram:

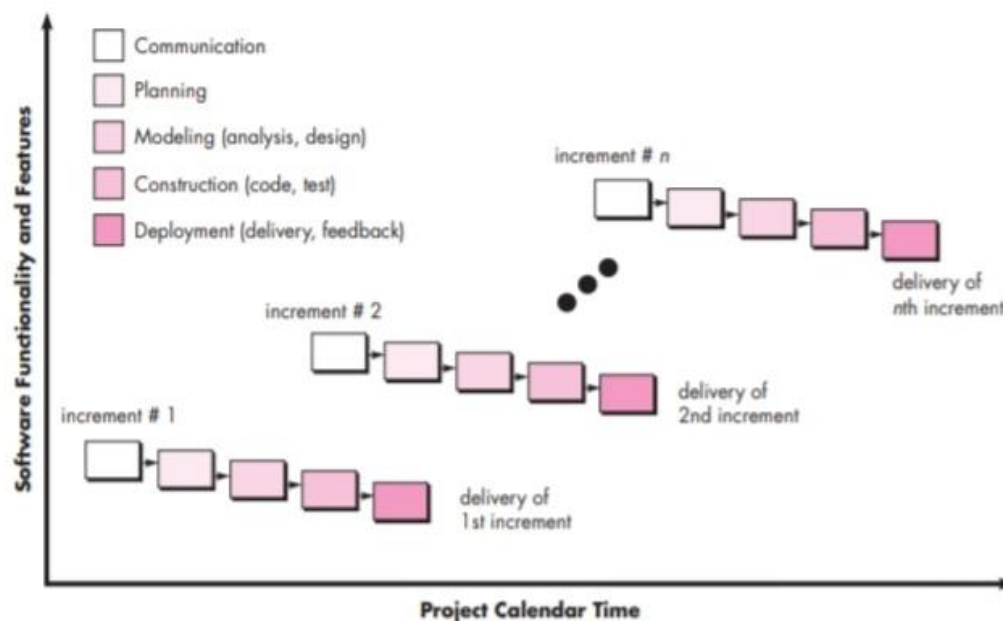


Fig.1.2 Incremental Model.

- **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.

8. Different phases of RAD model include:

- 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.

9. 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.

10. 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.
11. 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.

12. Diagram:

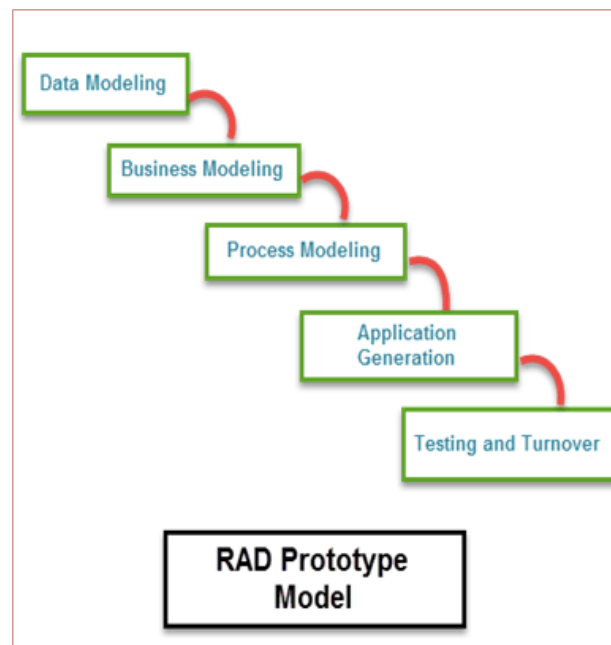


Fig.1.3 RAD Model.

- **Prototyping Model:**

In this model, a prototype of an 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.

13. Process of Prototyping:

1. Initial requirement identification.
2. Prototype development.
3. Review.
4. Revise.

14. Advantages:

1. The customer gets 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.

15. 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.

16. Diagram:

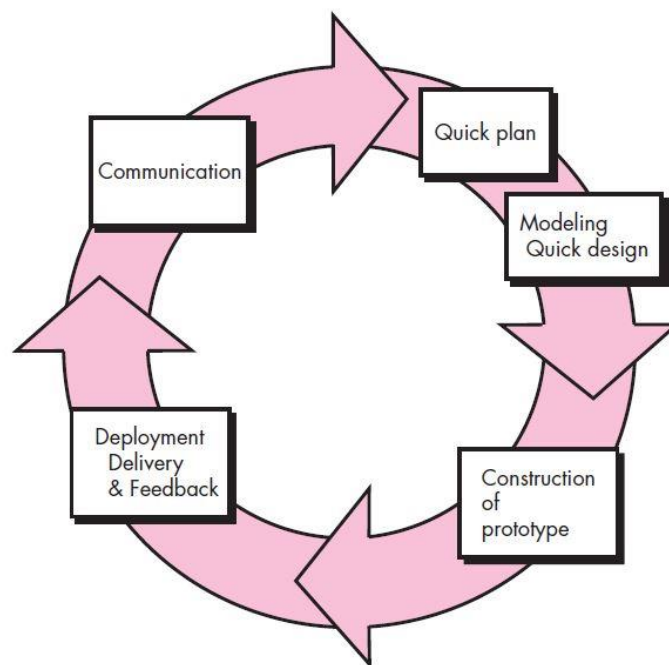


Fig.1.4 Prototype Model.

- **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 is:

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

17. 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.

18. 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.

19. 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.

20. Diagram:

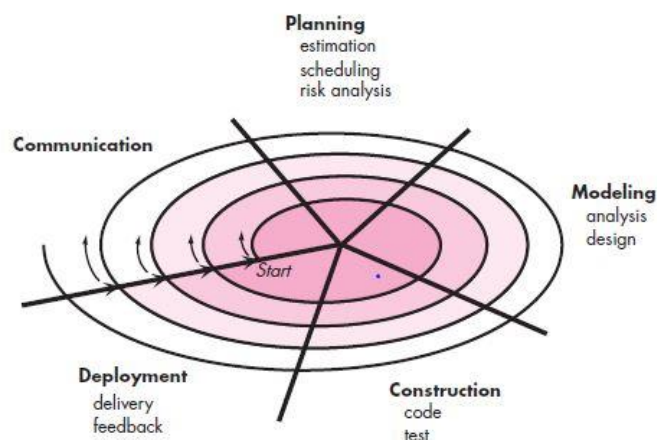


Fig. 1.5 Spiral Model.

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

- **Questions:**

- 1. Which model will you use?**

- I will use RAD (Rapid Application Development) model.

- 2. Why this model? Elaborate.**

- The RAD model focuses on project in small pieces, In RAD model, the larger projects are divided smaller projects and our project is for a huge government company, it will be a larger one that's why we will use RAD. Also, It is usually used where requirement is pre-defined and in our case the requirements are also predefined which are attendance and bio data management. RAD model is flexible and adaptable to the changes in case there are any to be made. Which will be useful for us if we need any future changes. In this model, scripts, high-level abstractions and intermediate codes are used so it will be easier to transfer the deliverable.



- **Conclusion:** In this experiment we learned about software development models and types of software development models, also in given scenario we applied suitable software development models and stated that why we used it.

|      |      |      |      |            |
|------|------|------|------|------------|
| (10) | (20) | (10) | (10) | Total (50) |
|      |      |      |      |            |

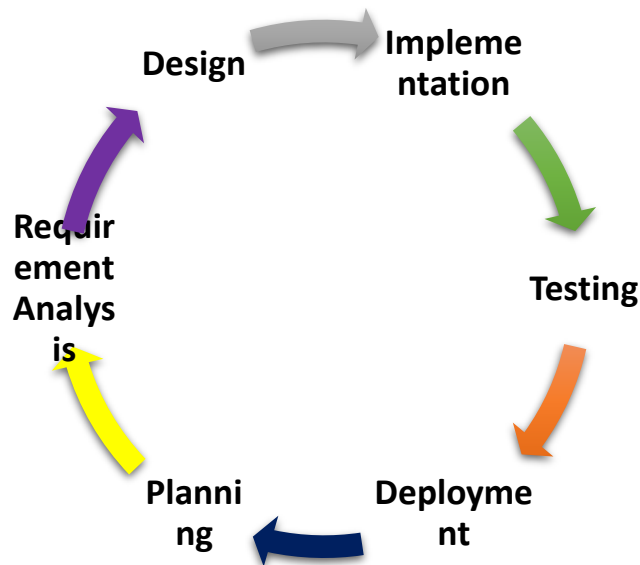
## **EXPERIMENT NO: 02**

**Aim:** 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**

Once the requirement analysis is done the next step is to define the documents the product requirement and get them approved from the customer or the market analysts. This is done through an SRS document which consists 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 are 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**

## **I. Planning**

- First and foremost, the features to be included in our application were discussed together by the team members. Also, the tools to be used in the making according to the hardware resources we all have and the open-source software's which can be used were made into consideration. The work each and every member will be doing were discussed according to their abilities.

## **II. Requirement Analysis**

- In Communication, the discussion with the stakeholders will take place about the requirements of our software that needs to be developed in software. As our requirements are discussed and planned, we made a list of the functional requirements, system and technical requirements. After communication, the SRS document which consist of all the product requirement to be designed is approved by the market analysts.

### **III. Modelling**

- In modelling phase, our team will work on GUI of software and some models like use case, state transition, activity diagram, etc. Software models are ways of expressing a GUI of software and its design. In our project modelling activity includes:
  - Analysing: - In Analysing, we analyse the need and working of each model. Also, a complete analysis of time and money and expected delivery of our final project.
  - Design: - GUI of our software will be made in this section, where all the modules are discussed and distributed to the team members.

### **IV. Construction**

- For making the application, Java will be used as the programming language. Page layout will be developed in Android XML. This project has been developed over the Android platform. We have used Android Studio for developing the project.

## **V. Deployment**

- The application and the SRS will be delivered to our client till the given deadline and feedback will be taken for future references.
- 
- **Questions:**
    - 1. Which process according to you works simultaneously and Why?**
      - According to me communication and modelling these two phases can work simultaneously as while communicating and getting the ideas and specific requirements from our customers we can concurrently make diagrams for our modelling phase.
    - 2. Which of the following is the most important phase in SDLC? why?**
      - In my opinion communication is the most important phase in SDLC as in if our communication with the stakeholders, customer, team members isn't good enough we won't be able to understand the requirements needed by customers and stakeholders about our application and hence the product won't live up to their expectations and needs of users. Also, if communications between the team members



doesn't go well it will end up in a negative working environment and result in failure. Hence the communication phase which is the first and foremost step should be done correctly.

### **3. From where do defects and failures in software testing arise?**

- Defects and Failures basically arise from errors in the specification, design and implementation of the software and system, errors in use of the system, environmental conditions, intentional damage, potential consequences of earlier errors

### **4. Which phase of the SDLC is known as the “ongoing phase” and why?**

- The Maintenance phase in SDLC is known as “ongoing phase”. As maintenance is the only phase which goes on even after the product's deployment to the customer. As our product is an application it needs maintenance. The maintenance phase involves making changes to hardware, software, and documentation to support its operational effectiveness. It includes making changes to improve a system's performance, correct problems, enhance security, or address user requirements.

- **Conclusion:** In this experiment we have identified the objectives and summaries outcomes for each SDLC phase, also in given scenario we identified the objectives and summaries outcomes for each SDLC of an Online Chatting Web Application successfully.

|      |      |      |      |            |
|------|------|------|------|------------|
| (10) | (20) | (10) | (10) | Total (50) |
|      |      |      |      |            |

## **EXPERIMENT NO: 03**

**Aim:** 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 user's 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 used to 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 especially 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 client's 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 Administration's 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 subjectifies 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 of 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 may 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 dependent 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 life 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 stakeholder's 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 conflict:

**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 forward 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

- **Scenario: Online Pharmacy System**

[illegible]

- **INTRODUCTION:** An online pharmacy is allowing user to consume information from anywhere through online. This application helps user to fulfil all their requirement while admitting themselves.

- Purpose:

- This software allows user to consume pharmacy from anywhere through online.
- It also allows user to fulfil their requirement.

- Scope:

- It is an online pharmacy which provides medicines to consumers through mail and shipping by using internet.
- It has cut down to chain of distributors and directly provides to user at less price.

- Definitions, Acronyms and Abbreviations:

- It is a web application that provides online medical service for people.
- Its acronyms are global as OPS (Online Pharmacy System).

- References:

- Object Oriented Software using UML pattern and java 3<sup>rd</sup> 2012.
- Cybermedicine: How computing empowers doctors and patients for better health care.

- Overview:

- It is built in order to replace manual based system.
- This management system is supposed too efficient, useful and affordable.
- It implements easily by pharmacy manager.

- **OVERALL DESCRIPTION**

- **Product Perspective:**

- This online pharmacy system is a standalone system. It has fulfilled all the stakeholder's requirement as per need.
- As a hardware, computer is used to run this software in medical stores.
- As per software interface, it includes medicine stocks, customers info, billing data in databases.

- **Product Function:**

- Stock Records Management System.
- Customer Information and Billing system.
- Sale and supplier management system.

- **User Characteristics:**

- Education about some technical terms – little.
- Knowledge about computer and Knowledge experience – little.
- Some knowledge about operating computer.

- **Constraints**

- A proper operating system target platform – chrome, Firefox.
- Its database will support less than 1000 columns.
- It will be required internet to store all the data into databases.
- It must ensure that high standards of service and care.

- **Assumptions and Dependencies**

- It does not need to store all generated reports.
- All the customers critical information are not necessary to include in it.

- **SPECIFICATION REQUIREMENTS**

- External Interface:
  - It has a hardware component which include RAM, hard disk, printer and swipe machine and other computer required components.
  - Also, JDK and SQL/MySQL as a software component.
- Functions:
  - Searching for required medicines as per need of customer. Stock Report.
  - Suppliers or customers information.
  - Online pharmacy though various portals.
- Performance Requirement:
  - Pentium II or more than that processors.
  - 512mb or more than that of RAM.
  - 800\*600 of screen revolution display/monitor
- Logical Database Requirement:
  - These software have to access to the computers storage or databases.
  - Also, if there's an external storage, it has to allow software to access it.
- Design Constraints:
  - This software could be web-based so every user can go through it by browsing, so they don't require any devices or software.
- Software System Quality:
  - Maintainability
  - Flexibility
  - Operability
- Object Oriented Models:
  - The online pharmacy system can be understood by diagram and relationship between classes which are used in software.

- **Appendices:**

- MySQL – to store information of various objects as a database.
- RAM – Random Access Memory.
- JDK – Java Development Kit.
- IDE – Integrated Development Environment

□ **Index:**

1. Functions
2. Operability
3. Overview
4. Performance Requirement

- **Conclusion:** We have design Software Requirement Specification (SRS) document for Online Pharmacy System. Considering Online Pharmacy System to be developed in any technology as a software Architect or project Manager. In this experiment we acknowledged all the components used in SRS of Online Pharmacy System and successfully prepared a document on Online Pharmacy System.

|      |  |      |      |      |            |
|------|--|------|------|------|------------|
| (10) |  | (20) | (10) | (10) | TOTAL (50) |
|      |  |      |      |      |            |



## **EXPERIMENT NO: 04**

**Aim:** 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 is usually expressed as a statement.

- **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 requirement 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.**

- **Questions:**

- 1. What are the Functional Requirements for above scenario?**

Storing information of employers and employees, updating their information in the management system, general report on working system, preparation of employer's reports regarding to salary attendance and biodata, deleting and editing the information of employers and employees as well as other necessary information about the company.

- 2. What are non-functional Requirements for above scenario?**

- **Performance Requirements:** Performance of the system should be fast and accurate. System should handle all the expected and unexpected errors. System should be able to handle a large amount of data.
- **Safety Requirements:** System must have two servers, one of them will be used as main server and one will be used as backup server.
- **Security Requirements:** Employee authentication and verification with their unique user ID and employee ID. System should have proper accountability. System also have Only Administrator Access to manage all the information of their employees and employers.

### 3. What are Benefits of Non-functional Requirements?

- Non-Functional Requirements ensures the software system should follow legal and compliance rules.
- Non-Functional Requirements also ensures the reliability, availability and performance of the software system.
- Non-Functional Requirements help in formulating security policies of the software system.

### 4. Differentiate between Functional Requirements and Non-functional Requirements?

| Parameters     | Functional Requirements                            | Non-Functional Requirements                           |
|----------------|--|---|
| Requirement    | It is mandatory.                                   | It is non-mandatory.                                  |
| Capturing type | It is captured in use case                         | It is captured as a quality attribute.                |
| End-result     | Product feature.                                   | Product Properties.                                   |
| Objective      | Helps to verify the functionality of the software. | Helps to verify the Performance of the software.      |
| Area of Focus  | Focuses on user requirements.                      | Concentrates on the use's expectation and experience. |
| Documentation  | Describe what the product does.                    | Describes how the product works.                      |
| Product Info   | Product features.                                  | Product properties.                                   |

- **Conclusion:** In this experiment, we learned about requirements of software system and classified the requirements into functional and non-functional requirements. Also using the given scenario, we have successfully identified and functional and non-functional requirements of a software system which is used for managements of salary attendance ad biodata of employees for a huge government company. We have also learned benefits of non-functional requirements and differentiated the functional and non-functional requirements.

| (10) | (20) | (10) | (10) | TOTAL |
|------|------|------|------|-------|
|      |      |      |      |       |

## **EXPERIMENT NO: 05**

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

### **Theory:**

- **What is a USE case diagram?**

In the Unified Modelling 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.

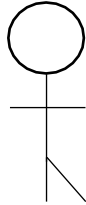


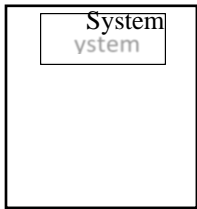
- **When to apply use case diagrams**

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

### 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.

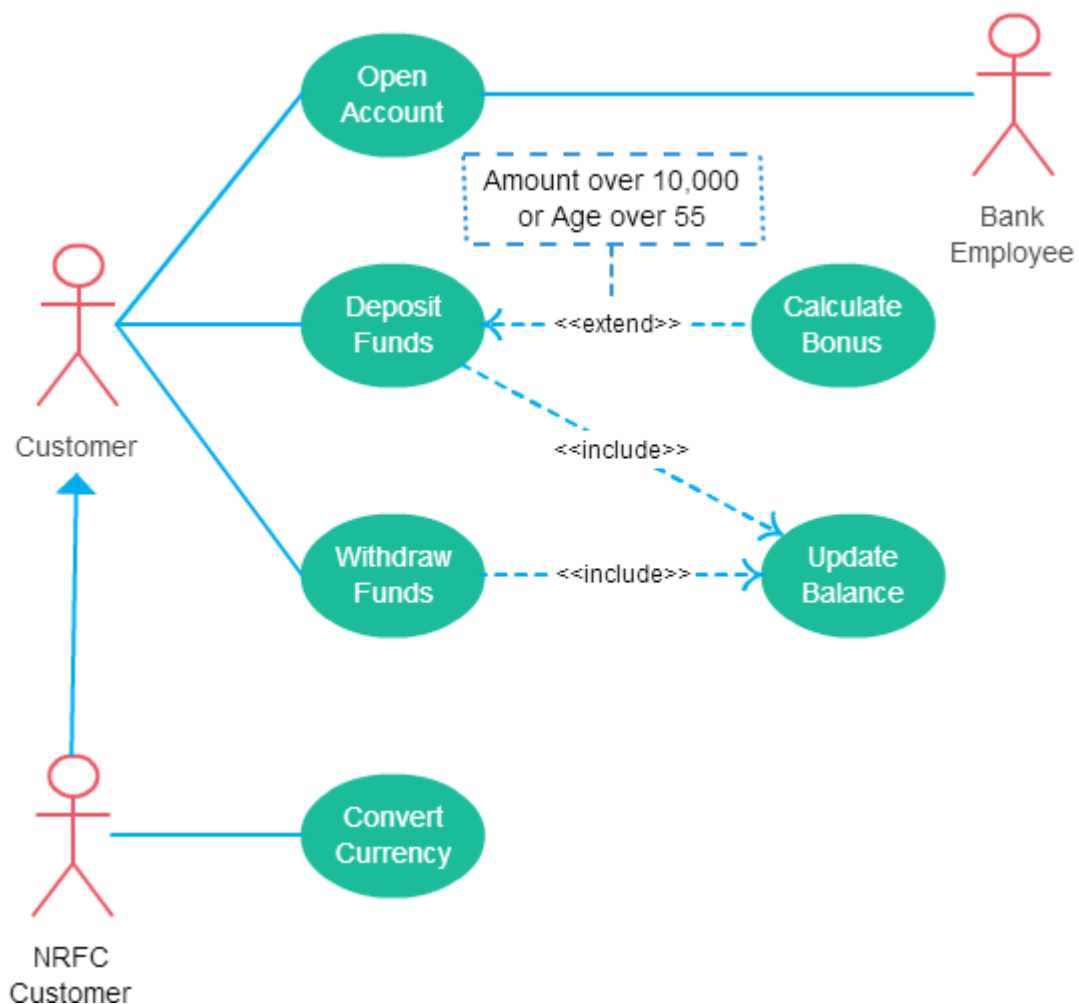


Fig. Example of use case diagrams

- **Scenario: Online Banking System**
- **Questions:**

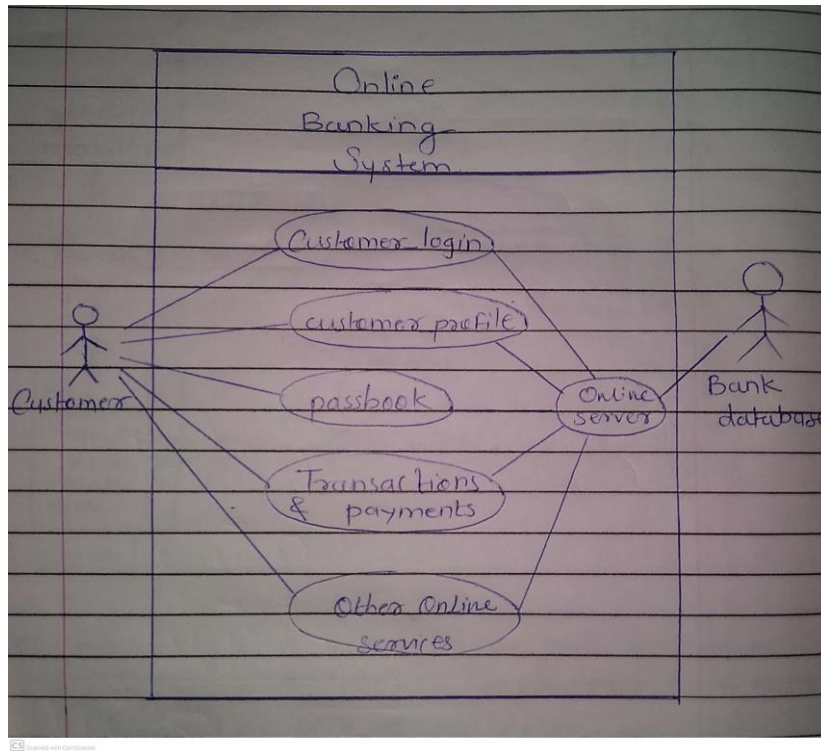
### 1. What is the Importance of Use Case Diagrams?

- Use Case Diagram is used to gather the requirements of a system.
- Use Case Diagram is used to get an outside view of a system.



- It is used to identify the external and internal factors influencing the system.
- It shows the interaction among the requirements of the of the software system.

## 2. Draw a USE case diagram for given scenario?



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

- No, The Use case diagram and functional requirements are different. The main difference is that use case diagrams are a graphical representation of the systems requirements, whereas functional requirements are in text form. Use cases can also have text but the main focus is on the diagram itself, whereas in functional requirements the focus is on the written text. Also, Functional requirements are operation of programs and defined as system in software development, whereas Use case diagram specify the events of the system and their flow.

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

- The flow of events and activities of use case diagram can also be modelled by using an activity diagram, because in the use case diagram the data flow in the system shows with the help of flow activity part which is also used in activity diagram to show the flow of data in the system. By using that part, both the diagrams show the flow of data or sequence.

- **Conclusion:** In this experiment we have learned about use case diagram and when to use and apply use case diagram also how to create a use case diagram. Using given scenario, we draw a use case diagram successfully and also learned its importance.

| (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 Modelling for System Integration).

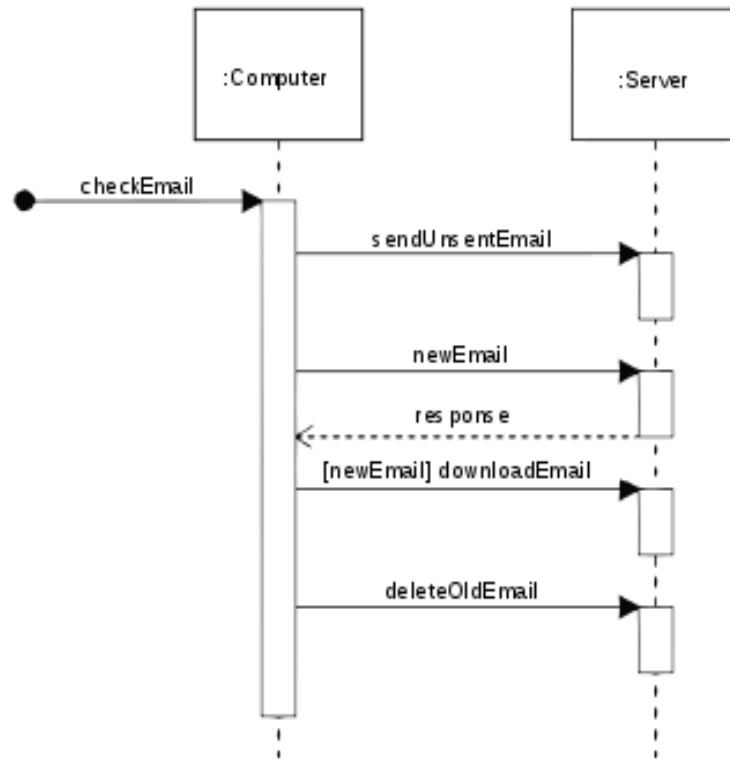


Fig: Example of sequence diagram

- **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 behaviour of a system. This behaviour is analysed 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".

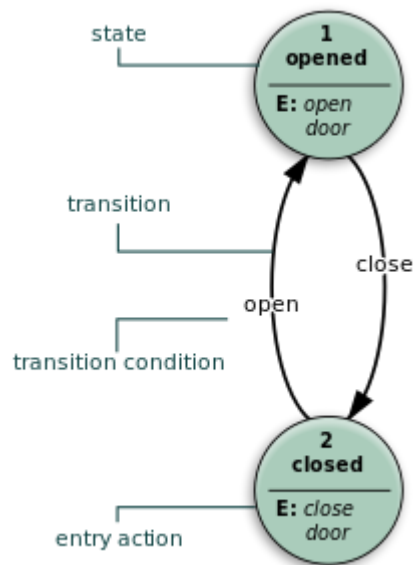


Fig: Example of State diagram

- **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.

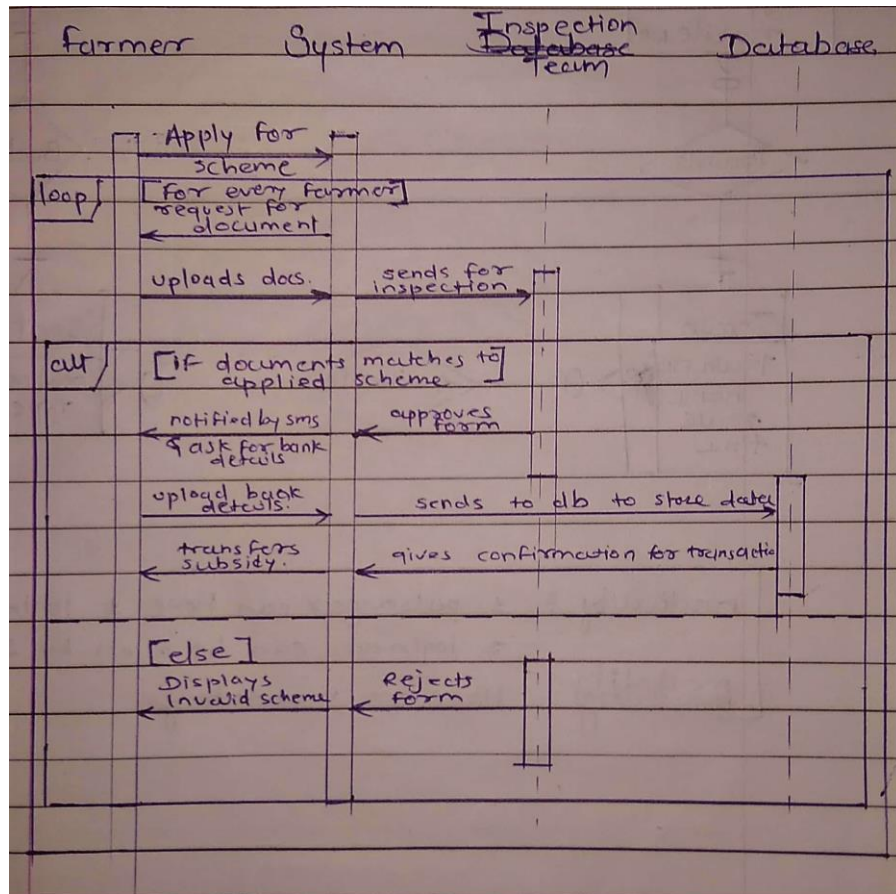
- **Scenario:**

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

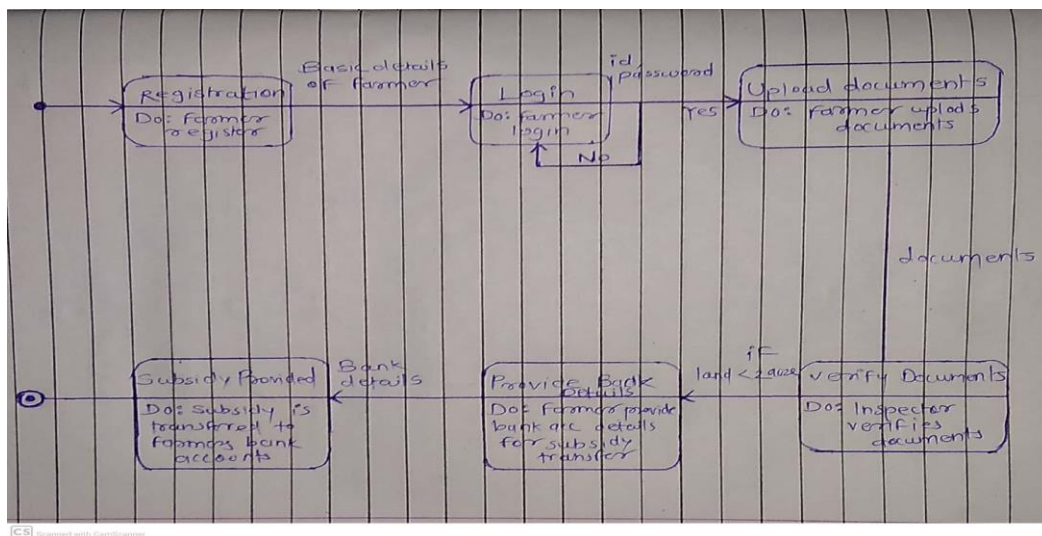
- a. Farmer needs to register themselves on a provided website.
- b. Farmer needs to upload required documents.
- c. Inspection team inspects farmer document and land.
- d. Government successfully transfers the subsidy in farmer's bank account.

- Questions:

1. Draw Sequence diagram for given scenario.



2. Draw State diagram for given scenario.





### 3. Comparison between sequence diagram and state diagram.

| Sr. No | Sequence Diagram   | State Diagram   |
|--------|--|---|
| 1.     | A sequence diagram shows object interactions arranged in time sequence.  | A state diagram is a type of diagram used in computer science and related fields to describe the behavior of systems.   |
| 2.     | Sequence diagrams are sometimes called event diagrams or event scenarios.  | State diagram are sometimes called as Harel state chart or a state machine diagram  |
| 3.     | A sequence diagram describes the events for a single interaction across all objects involved   | A state diagram describes all events and state and transition for a single object   |
| 4.     | It Used to model and visualize the logic behind a sophisticated function, operation or procedure, They are also used to show details of UML use case diagrams. Used to understand the detailed functionality of current or future systems. | 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. |

- **Conclusion:** In this experiment, we have learned about sequence diagram and high-level sequence diagram also about state diagram and its purpose. Using this information about sequence and state diagram, we have successfully drawn sequence and state diagram for given scenario and also compare sequence diagram and state diagram.

| (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 diagram to display the contents of a data store.

- **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.

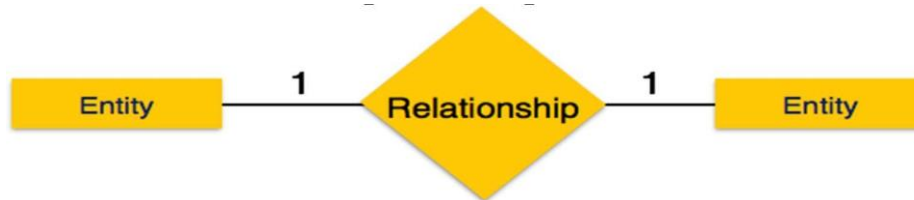


- **Connecting lines**, solid lines that connect attributes to show the relationships of entities in the diagram.

- **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

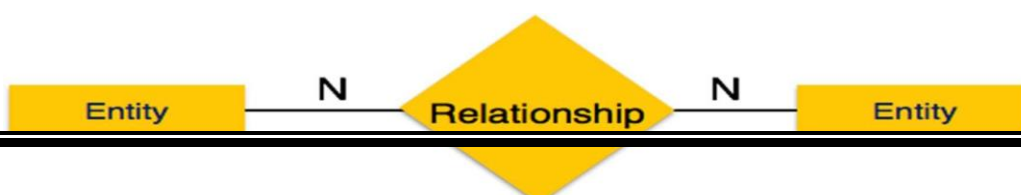


relationship.

- **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

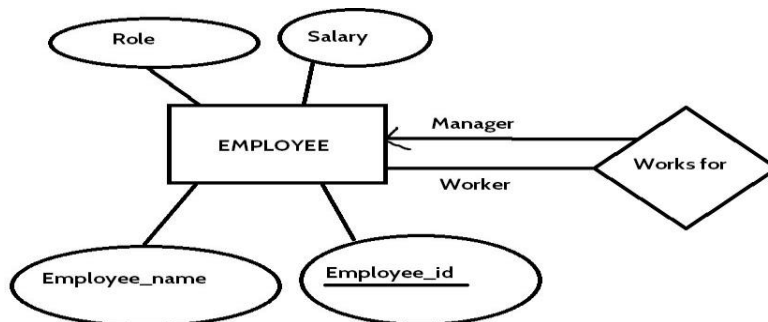


instance of an entity on the right can be associated with the relationship. It depicts many-to-many relationship.

- **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 data flow 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 (Last Name, FirstName, SS#, Tel #, etc.)
- ☐ Order info (Ordered, Item#, Order Date, Customer, 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 is 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:**

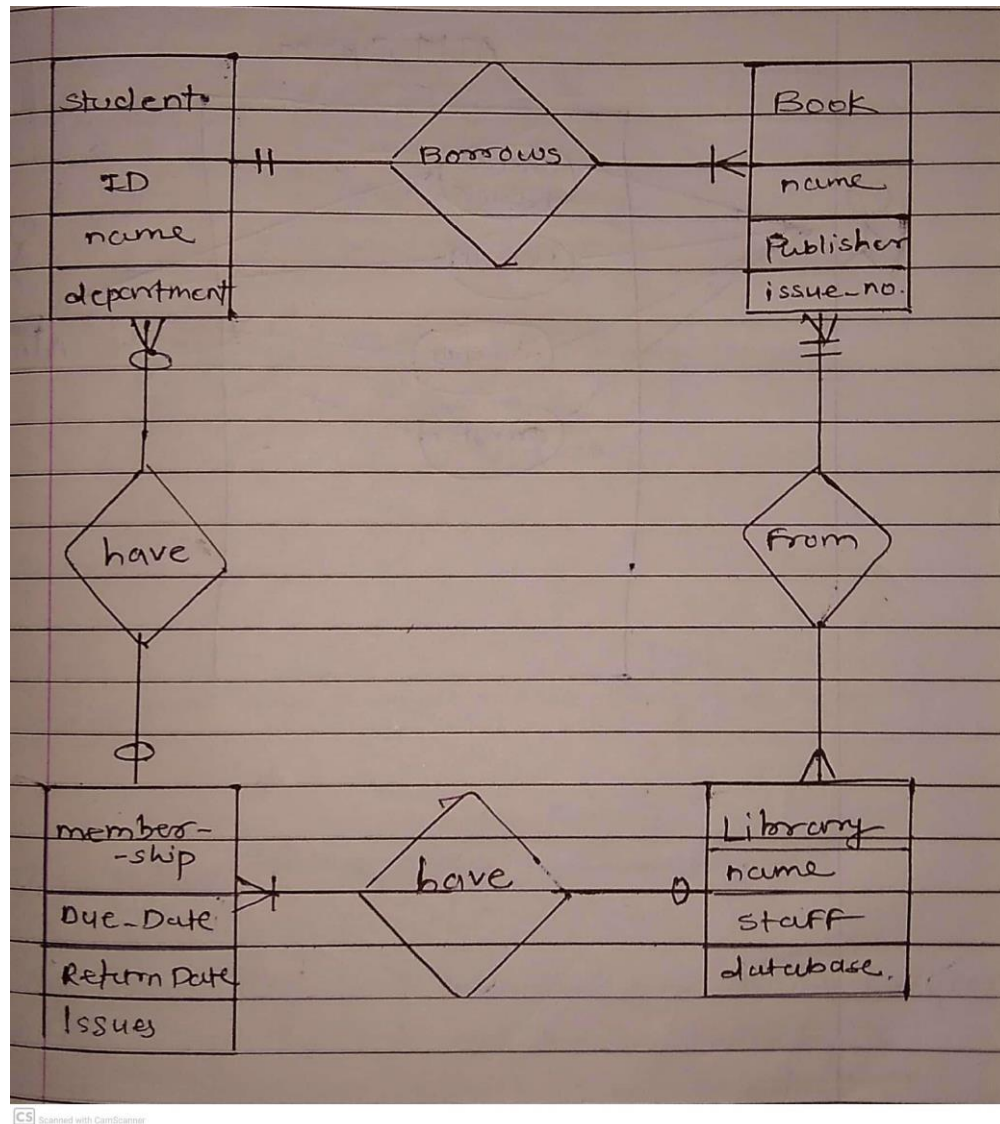
There are a number of advantages of using Data Dictionary in computer system analysis and design. The advantages are: consistency, clarity; reusability; completeness; increase in sharing and integration; and ease of use for the developer.

- **Scenario:**

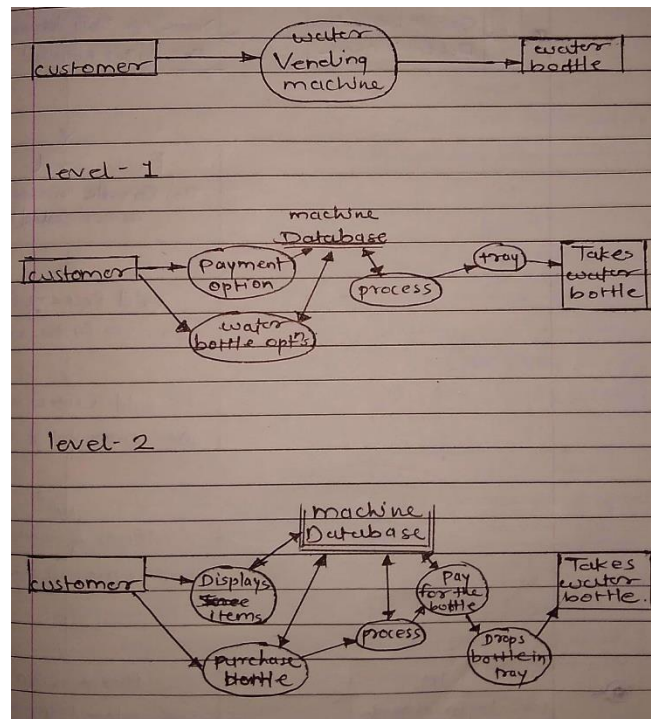
1. **Library Management System**
2. **Water Vending Machine**

- Questions:

1. Draw the ER diagram for given scenario 1?



## 2. Draw DFD diagram for given scenario 2?



## 3. What is the objective of maintaining data dictionary?

Data dictionaries are used to provide detailed information about the contents of a dataset or database, such as the names of measured variables, their data types or formats, and text descriptions. A data dictionary provides a concise guide to understanding and using the data.

- **Conclusion:** In this experiment, we have learned about entity-relationship diagram and its uses and relationship and symbols also, its advantages and disadvantages. Along with ER-diagram we also learned about data flow diagram and its different types. We also draw a ER diagram on library management system and a DFD diagram on water vending machine 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 are similar to other four diagrams. It captures the dynamic behaviour 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.

# Activity Diagram

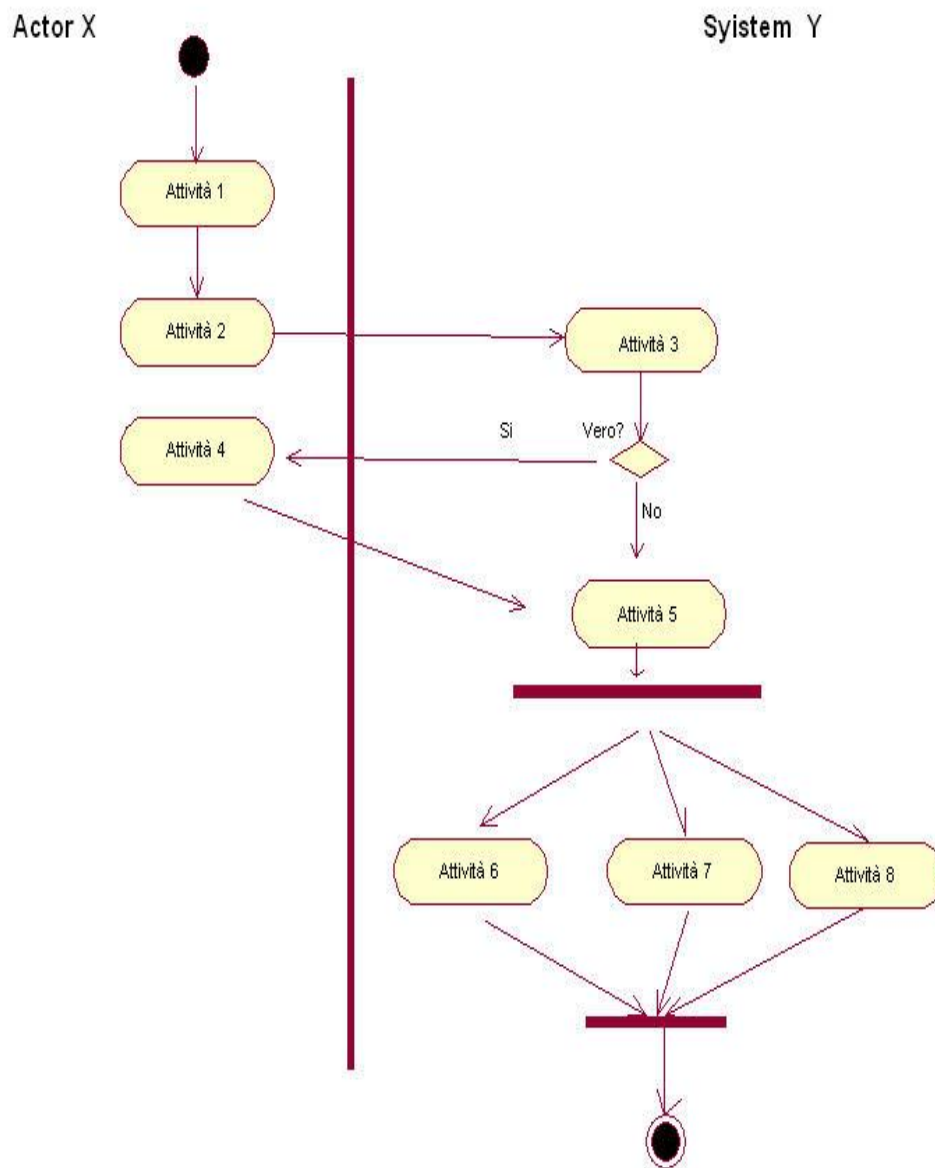


Fig: Activity Diagram

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 –

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

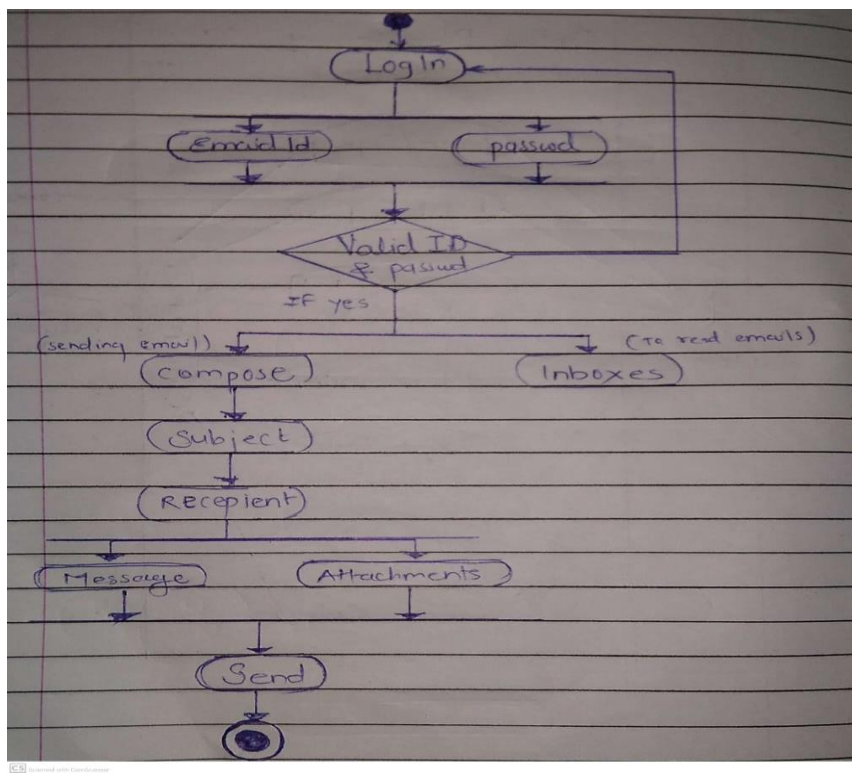
- **Scenario: Email account opening and managing system**

- **Questions:**

**1) When to Use Activity Diagram?**

- Activity diagram is used when we want to show message flow from one activity to another, for the purpose of functional modelling, describe the general sequence of actions for several objects and use cases, represent the execution of the process.

**2) Draw activity diagram for given scenario.**



### 3) Comparison between activity diagram and sequence diagram.

| Parameter         | Sequence Diagram   | Activity Diagram  |
|-------------------|--|---|
| <b>Definition</b> | The Sequence diagram represents the UML, which is used to visualize the sequence of calls in a system that is used to perform a specific functionality. The Sequence diagram shows the message flow from one object to another object. | The Activity diagram represents the UML, which is used to model the workflow of a system. The Activity diagram shows the message flow from one activity to another. |
| <b>Main focus</b> | Main focus is the interaction between different objects over a specific period of time.  | Main focus is the flow of activities.   |
| <b>Helps</b>      | Helps to visualize the sequence of calls in a system to perform a specific functionality.  | Helps to model the workflow a system  |
| <b>used</b>       | Sequence diagram is used for the purpose of dynamic modelling. Sequence diagram is mainly used to represent the time order of a process.   | Activity diagram is used for the purpose of functional modelling. Activity diagram is used to represent the execution of the process.                               |

- **Conclusion:** In this experiment, we have learned about activity diagram and its purpose and use of activity diagram. Also using given scenario, we draw an activity diagram on email opening and management system successfully.

| 10 | 20 | 10 | 10 | Total |
|----|----|----|----|-------|
|    |    |    |    |       |

## **EXPERIMENT NO: 09**

**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: -**

Well-designed 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.

- **Scenario: Home security system with camera and sensors**

- **Questions:**

- 1. Which Principles are being violated with the given scenario?**

- In the given scenario, the design should be structured to degrade gently, even when aberrant data, events, or operating conditions are encountered principle are being violated.

## **2. How it is violated with the given scenario.**

- In the given scenario, the design should be structured to degrade gently, even when aberrant data, events, or operating conditions are encountered principle are being violated, because in the home security in which we are going to use hardware components such as cameras, sensors, monitors and electrical wires may have chances to get damaged by uncommon conditions and in any rare situation like disturbance in flow of electric current, line conditioner and moist temperature. But this principle state that the software design should not be bomb, it should be designed to accommodate unusual circumstances and must terminate the processing in the refined manner which is not applicable for the given scenario. The failures in home security with camera and sensors would not be able to assist these uncommon and unexpected situations and conditions.

## **3. Design is not coding; Coding is not design relate it with the scenario.**

- In any software development process, designing and coding are two different terms. If we relate it with given scenario, you will need to know about what type of devices and hardware are going to use in the system when you work on the designing of the system. And when you start working on coding of the system you will need to know about code which need to be developed on the system and which type of system will work on the code you are going to use and to get the proper results of the working of system you need to performed both of them on the system.

- **Conclusion:** In this experiment, we have learned the design principles of the software engineering in which we went through 10 design principles and get to know that to get quality factors for our system we need these 10 design principles. We also identified the principles which are being violated with given scenario which was home security with cameras and sensors.

| (10) | (20) | (10) | (10) | TOTAL |
|------|------|------|------|-------|
|      |      |      |      |       |



## **EXPERIMENT NO: 10**

**Aim:** Mini Project on Online Book Store.

### **Introduction:**

The online book store is an application software made to help students as well as a majority of book readers for borrowing and renting reference books, guides and novels online. At the time of this pandemic where physical contact needs to be minimized whereas classes are conducted online students are not able to access college library for guidance. Hence the application will be saving a lot of effort and time.

### **Process:**

#### **1. Communication**

- The project was given by Prof.Lokesh Bhadekar to our team members (total 6 members).
- The topic was to make an Online book store
- The basic features as suggested were to be able to read, borrow or rent e-books online hence saving time and efforts.

#### **2. Planning**

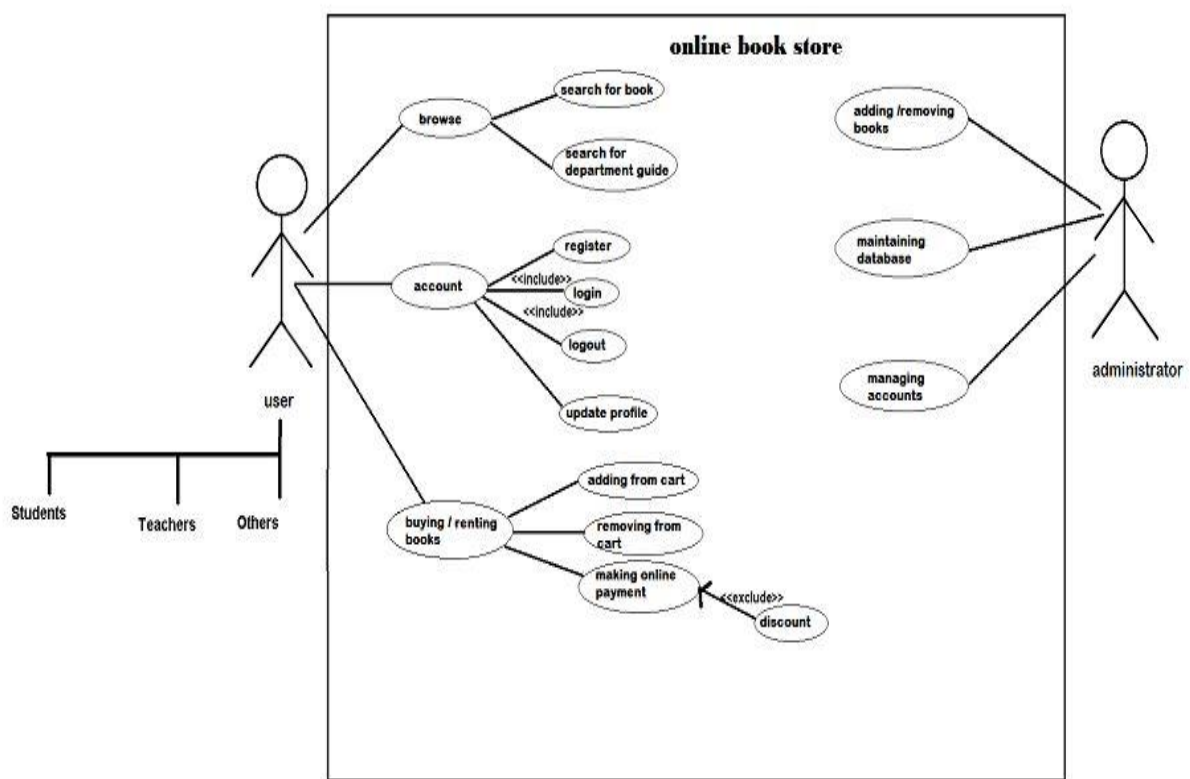
- First and foremost, the features to be included in our application were discussed together by the team members.
- Also, the tools to be used in the making according to the hardware resources we all have and the open-source

software's which can be used were made into consideration.

- The work each and every member will be doing were discussed according to their abilities.

### 3. Modelling

#### Use case diagram



use case diagram

### 4. Construction

- For making the application, Java was used as the programming language.
- Page layout was developed in Android XML.
- This project has been developed over the Android platform.
- We have used Android Studio for

developing the project. – As for testing, it was done manually by team members by performing different actions on the application in order to find any errors and debug them beforehand.

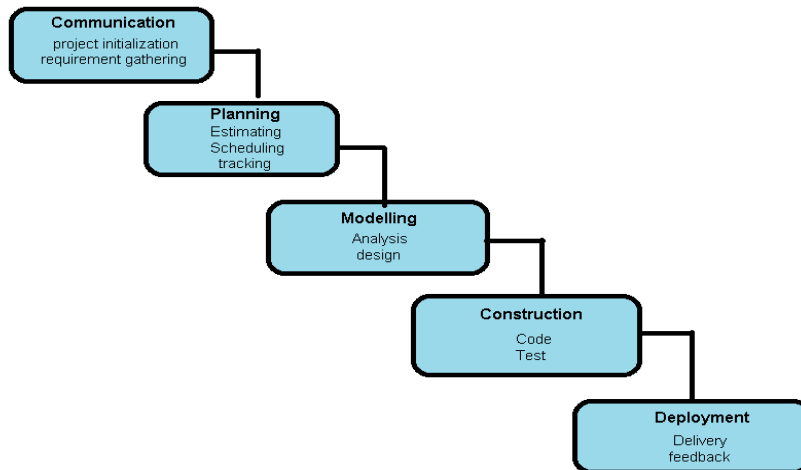
## **5. Deployment**

- The application is to be submitted to our project guide by 11 January 2021, Monday. The project report will be submitted on schoology app by Monday in a pdf format. The application's layout and working will be presented on the same day.

## **Process Models**

### **Waterfall Model**

- Our project requires a process model that works linearly as we will be working step by step.
- The project starts with communication and ends with deploying it to the project guide.
- We will be moving on to the next step only after completing the previous one.
- Also, here the user requirements were fixed. Hence waterfall been the one for us.
- Given below is the Waterfall model's diagram.



## **Project Management Concepts**

### **People**

- The team leader was Megha Shivhare managed the team and defined the significant tasks for other members.
- The coding work for the application was done by Bhavin Patil.
- The documentation work for report and SRS and other significant tasks were done by Kunal Warade, Anuja Pakkide, Tanisha Sahare and Toshil Kumbhalkar
- The customer for our project was our project guide (Prof. Lokesh Bhadekar)
- As for the end users the product focuses on college students.

### **Product**

### **Software Scope**

- **Information Objectives** - Online book store is an application software. The main objective of the online

book store is to manage the details of books, customer payments and bills. It manages all the information about books stocks, books bill etc.

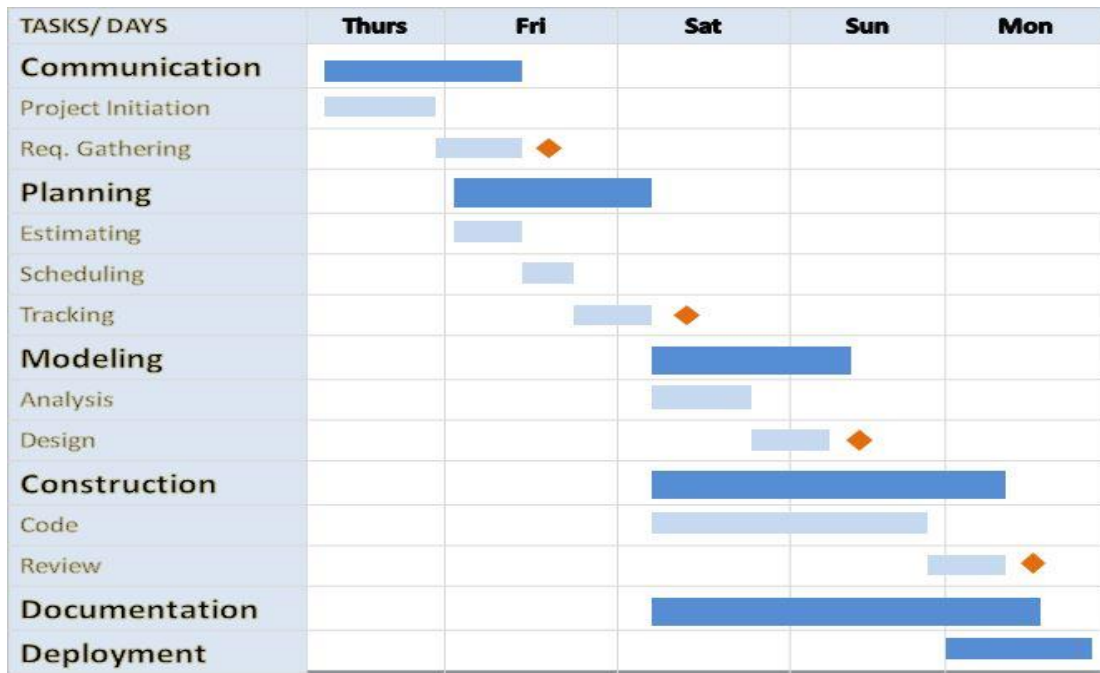
- **Function and Performance** - Through our search functionality the customer can search for a book by its title or author, later can add to the shopping Cart and finally purchase using credit card or any online transactions.

## **Process**

- As for process the waterfall model was used which is already explained above.
- For **process decomposition**, the customer (our project guide) provided us everything clearly and hence there were no clarification issues.
- A scope for the application was developed by taking customer's opinions into consideration.

## **Project**

- Proper decisions were made by defining proper jobs to the team members.
- We focused on making the application simple, understandable and easy to use.
- For tracking our project's process, we have used the Gantt chart given below:



## W5HH Principle

### 1. Why is the system being developed?

- The system is been developed in order to fulfill the need of wide range of books at one stop. The system will be providing e-books online saving our time and efforts. Also, it will be ensuring safety as no physical contact will take place and at such times this becomes it's one of the best features.

### 2. What will be done, by when?

- The key tasks are already defined: Communication, Planning, Modeling (Use case diagram, activity diagram and state diagram), Documentation (making of project report and SRS), coding and deployment. The time to be taken was decided already through Gantt chart. Duration for each of these were defined as follows:

| Tasks         | Duration (in hrs.) |
|---------------|--------------------|
| Communication | 4                  |
| Planning      | 8                  |
| Modeling      | 2                  |
| Documentation | 10                 |
| Coding        | 10                 |
| Deployment    | 2                  |

### **3. Who is responsible for a function?**

- The tasks were defined to different members according to their capabilities.
- The team leader was Megha Shivhare managed the team and defined the significant tasks for other members.
- The coding work for the application was done by Bhavin Patil.
- The documentation work for report and SRS and other significant tasks were done by Kunal Warade, Anuja Pakkide, Tanisha Sahare and Toshil Kumbhalkar.

### **4. Where are they organizationally located?**

- The customer for our project is Prof Lokesh Bhadekar that is our project guide from our college.
- As it is a college project it wouldn't hold any end users in real but the app focuses on book readers and students from different almost all age groups.

## **5. How will the job be done technically and managerially?**

- On the Technical front software tools like Android Studio were used and for the programming language Java was used.
- Managerially things were managed by team leader by assigning different tasks and monitoring them.

## **6. How much of each resource is needed?**

- Human Resources that are the team members and customer will be needed for the making of the software and its successful communication and deployment.
- Hardware resources include PC, with enough memory space and 8 GB RAM will be needed.
- Software needs include Android platform for testing the work of application at times and Android studio for making of the application.

## **Software Risks- RMMM**

### **Risk - Poor Productivity/Quality**

**Mitigation:** As schedule is tight there may be chances of compromising with the quality of software to avoid this quality is maintained at every stage by the team leader also the work is properly distributed on the planning phase.

**Monitoring:** By performing basic testing and keeping an eye on the product quality at every stage the team leader monitors this risk.



**Management:** In case the risk occurs and the software doesn't stand out in terms of quality the team leader can order some changes to be made to make the quality better and negotiate with the customer at some points.

### **Risk - Wrong estimation**

**Mitigation:** The tasks and the person performing them were judged properly in order to take out a rough estimation of the time it'll need.

**Monitoring:** The estimations are be monitored using a variety of different softwares and charts also we have used the Gantt chart.

**Management:** In that case we'll try to cope up with them by speeding up our work speed as well as by performing activities in a parallel manner so that we can complete the tasks till the due date.

### **Risk –Incompleteness**

**Mitigation:** To avoid late delivery the team leader makes the Gantt chart to track the project flow.

**Monitoring:** For monitoring the application again the team leader uses the Gantt chart to track the project.

**Management:** In case the application is not made complete till due date (Monday) we will be presenting our application till it is completed and our ideas on how it will work when completed. The report in case if not completed will be submitted on schoology tough as late submission.

### **Reusable Software Resources**

- As for reusability we will be using some modules and functions from a previous project we made.
- The previous project included the login and sign-up page which were to be included in this one also.
- The functions and modules from those logins and sign-up page were used here although changes were made in the theme and colours.
- Hence it resulted in a lot of decrease in the amount of time and effort required and so we could concentrate on other things.

# **Software Requirement Specification**

## **Title: Online Book Store**

### **Table of contents**

#### **1. Introduction**

- Purpose
- Scope
- Definitions , Acroynms and Abbreviations
- References
- Overview

#### **2.Overall Description**

- Product perspective
- Product functions
- User characteristics
- Constraints
- Assumptions and Dependencies

### **3. Specific Requirements**

- External Interfaces
- Functions
- Performance Requirements
- Logical Database Requirements
- Design Constraints
- Software System Quality Attributes
- Models

### **4. Appendices**

### **5. Index**

## **1. Introduction**

### **1.1 Purpose**

The system is built to help book readers to be benefitted by borrowing/renting e-books online also at a time like this where physical contact is not safe. On the other hand, this saves a lot of time and is efforts. Also, the

software is economically feasible for as e-books cost much less.

## **1.2 Scope**

The online book store is an one-stop destination for book lovers as well as for students who need to refer guides or reference books for studying purpose. A wide range of e-books are made available promoting safety as of no physical contact will be taking place. Also it will be helpful for students who are not able to access the college library as the colleges are yet not open.

## **1.3 Definitions, Acronyms and Abbreviations**

- **Data Entity:** An entity is a thing, person, place, unit, object or any item about which the data should be captured and stored in the form of properties, workflow and tables.
- **RAM:** Random Access Memory (i.e. main memory)
- **IDE:** Integrated Development Environment

## **1.4 References**

- Programming with Java by E. Balagurusamy.
- <https://www.javatpoint.com/>
- <https://developer.android.com/studio>

## **1.5 Overview**

The purpose of this SRS document is to provide a detailed overview of the online book store. It contains a general description of its working, how it is going to work and the technology used.

## **2. Overall Description**

### **2.1. Product Perspective**

The product is seen as a college project given to the team members for practical by Prof. Lokesh Bhadekar. The project will help us in learning the basic concepts of Software Engineering. Also we'll be implementing these concepts to understand them better.

### **2.2. Product Functions**

The product is capable of performing following functions

1. Login for customer
2. Logout Functionality
3. Customer registration
4. Costumer My account
5. Costumer can search for to get list of books
6. All available product category and
7. Costumer can see book details with images as well as his order history and order items
8. Costumer can add/delete products from cart with quantity
9. Costumer will be able to make a payment online

## **2.3. User Characteristics**

User must have basic skills and knowledge to operate the software and to work on it.

## **2.4. Constraints**

Android XML: Page layout has been designed in Android XML

Android: This project has been developed over the Android Platform.

Java: All the coding has been written in Java

Android Studio: We have used Android Studio for developing the project

## **2.5. Assumptions and Dependencies**

The product to work properly requires an Android system of version 5.0 or above.

# **3. Specific Requirements**

## **3.1. External Interfaces**

The hardware components include RAM, hard disk and other PC components. Also, it will require Android studio.

### **3.2. Functions**

1. Login for customer
2. Logout Functionality
3. Customer registration
4. Costumer My account
5. Costumer can search for to get list of books
6. All available product category and
7. Costumer can see book details with images as well as his order history and order items
8. Costumer can add/delete products from cart with quantity
9. Costumer will be able to make a payment online

### **3.3. Performance Requirements**

The system software and hardware requirements to work properly are a smartphone or any such device with basic features, enough memory space with an android system of 5.0 version or above.

### **3.4. Logical database requirements**

The logical database includes the data entities which include the user details (username, password, email id), book details (name, subject, author, in stock).

### **3.5. Design Constraints**

The product is an application software which can work properly on any android device.

### **3.6. Software System Quality Attributes**

#### **- Operability**

The input for various actions will be given through keyboard and touch on smartphones

#### **-Correctness**

Tests were done manually by the members have been performed by the members in order to check the accuracy and errors are debugged before delivery.

#### **-Flexibility**

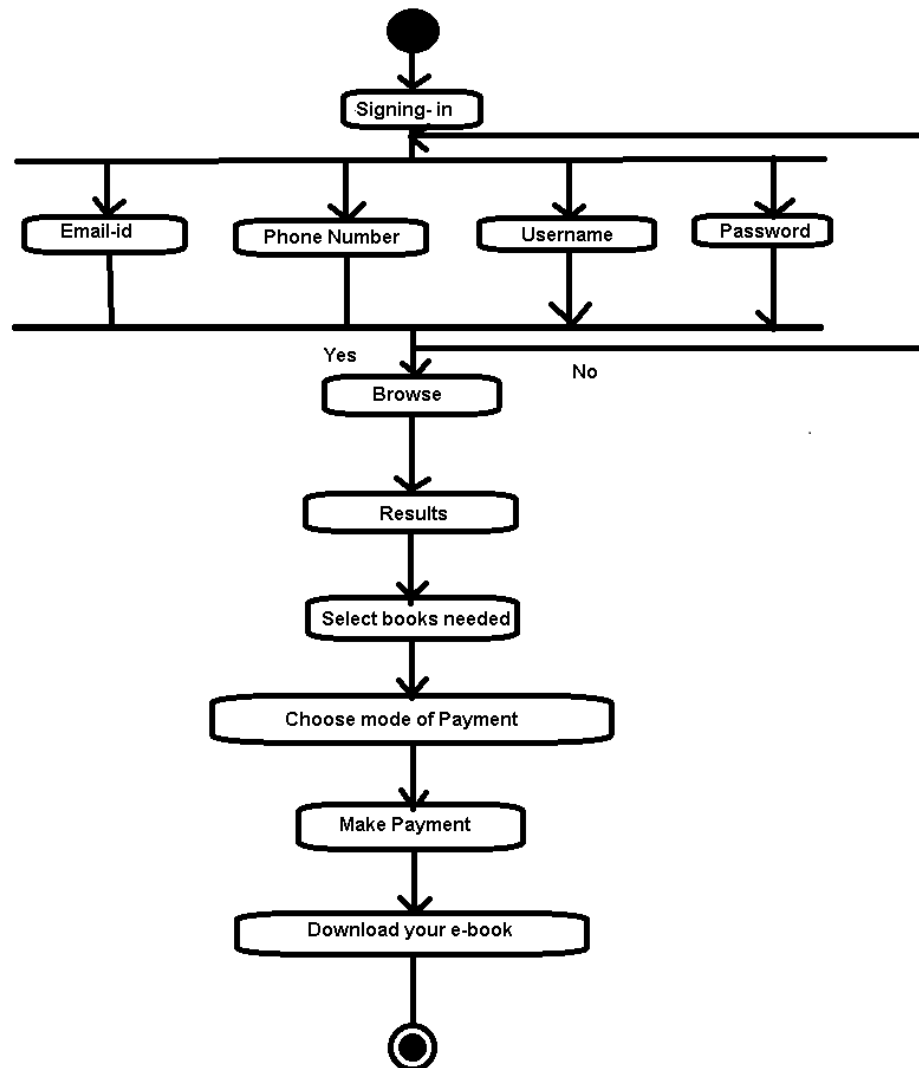
If the client wants, we are willing to make some minor changes after delivery.

### **3.7. Models**

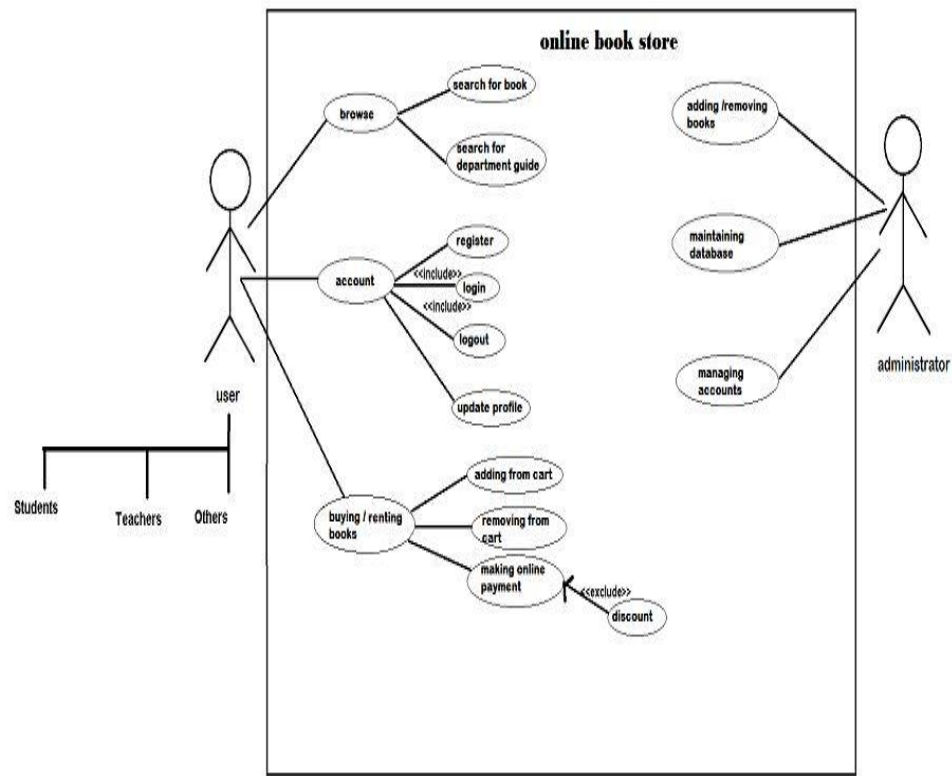
The online book store application software can be described by the use class diagram and other such models to show relationships between the classes that compose the system.



## Activity Diagram

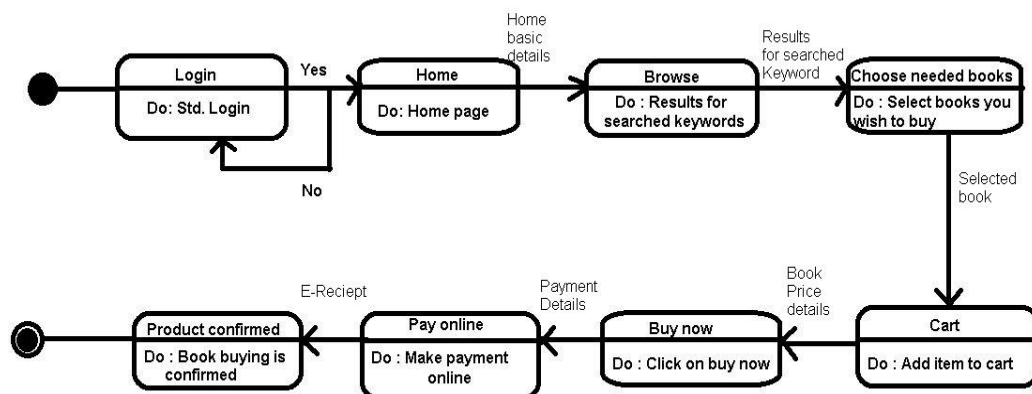


## Use Case Diagram



use case diagram

## State diagram



## 4.Appendices

**Data Entity:** An entity is a thing, person, place, unit, object or any item about which the data should be captured and stored in the form of properties, workflow and tables.

**RAM:** Random Access Memory (i.e., main memory)

**IDE:** Integrated Development Environment

## 5.Index

| Topic            | Page no. |
|------------------|----------|
| Functions        | 15       |
| Operability      | 17       |
| Overview         | 15       |
| Performance Req. | 17       |

- **Conclusion:** In this experiment, we have made a mini project on the software engineering using all the concepts of software engineering. We made a project report on online book store software, in which we have explained everything about our project. We have applied project management concepts, W5HHH principles, explained all the risks related to our project and also made a documentation on SRS of the project with all the required diagrams to our project.

| (10) | (20) | (10) | (10) | TOTAL |
|------|------|------|------|-------|
|      |      |      |      |       |