



Datta Meghe College of Engineering
Airoli, Navi Mumbai

DEPARTMENT OF COMPUTER ENGINEERING
ACADEMIC YEAR: 2023 – 24 (TERM – I)

List of Experiments

Course Name : SOFTWARE ENGINEERING LAB

Course Code : CSL501

Sr. No	Name of experiment	CO's Covered	Page No.	Date of Performance	Date of Submission	Marks & Signature
01	Application of at least two traditional process models	CSL501.1				
02	Application of the Agile process models.	CSL501.1				
03	Preparation of software requirement specification (SRS) document in IEEE format.	CSL501.2				
04	Structured data flow analysis.	CSL501.2				
05	Use of metrics to estimate the cost.	CSL501.3				
06	Scheduling & tracking of the project.	CSL501.3				
07	Write test cases for black box testing.	CSL501.5				
08	Write test cases for white box testing.	CSL501.5				
09	Preparation of Risk Mitigation, Monitoring and Management Plan (RMMM).	CSL501.6				
10	Version controlling of the project.	CSL501.6				
11	Assignment-1	CSL501.1, CSL501.2, CSL501.3				
12	Assignment-2	CSL501.4, CSL501.5, CSL501.6				
13	Mini Project	CSL501.1				

This is to certify that Mr. / Miss _____ of _____ Roll No. _____ has performed the Experiments / Assignments / Tutorials / Case Study Work mentioned above in the premises of the institution.

Prof. Jayant Sawarkar
Practical Incharge



**DATTA MEGHE COLLEGE OF ENGINEERING, AIROLI, NAVI
MUMBAI
DEPARTMENT OF COMPUTER ENGINEERING**

Institute Vision : To create value - based technocrats to fit in the world of work and research

Institute Mission :

- To adopt the best engineering practices
- To empower students to work in the world of technology and research
- To create competent human beings

Department Vision : To provide an intellectually stimulating environment for education, technological excellence in computer engineering field and professional training along with human values.

Department Mission :

M1: To promote an educational environment that combines academics with intellectual curiosity.

M2: To develop human resource with sound knowledge of theory and practical in the discipline of Computer Engineering and the ability to apply the knowledge to the benefit of society at large.

M3: To assimilate creative research and new technologies in order to facilitate students to be a lifelong learner who will contribute positively to the economic well-being of the nation.

Program Educational Objectives (PEO)

PEO1: To explicate optimal solutions through application of innovative computer science techniques that aid towards betterment of society.

PEO2: To adapt recent emerging technologies for enhancing their career opportunity prospects.

PEO3: To effectively communicate and collaborate as a member or leader in a team to manage multidisciplinary projects.

PEO4: To prepare graduates to involve in research, higher studies or to become entrepreneurs in long run.

Program Specific Outcomes (PSO)

PSO1: To apply basic and advanced computational and logical skills to provide solutions to computer engineering problems.

PSO2: Ability to apply standard practices and strategies in design and development of software and hardware-based systems and adapt to evolutionary changes in computing to meet the challenges of the future.

Program Outcomes as defined by NBA (PO)

Engineering Graduates will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

DATTA MEGHE COLLEGE OF ENGINEERING, AIROLI

DEPARTMENT OF COMPUTER ENGINEERING

COURSE NAME: SOFTWARE ENGINEERING LAB

COURSE CODE: CSL501

YEAR OF STUDY: T.E., SEMESTER: V

COURSE OUTCOMES

CSC502.1	Identify requirements & assess the process models.
CSC502.2	Able to analyze and specify software requirements from various Stakeholders in a specific format.
CSC502.3	Plan, schedule and track the progress of the projects.
CSC502.4	Design the software projects.
CSC502.5	Do testing of software project.
CSC502.6	Identify risks; manage the change to assure quality in software projects.

DATTA MEGHE COLLEGE OF ENGINEERING
DEPARTMENT OF COMPUTER ENGINEERING
ACADEMIC YEAR 2023-24 (TERM I)
SUBJECT: SOFTWARE ENGINEERING LAB (CSL501)
SEM: V DIV : A & B
RUBRICS FOR GRADING EXPERIMENTS

Rubric Number	Rubric Title	Criteria	Marks* (out of 15)
R1	Timeline	On-time	5
		Delayed by not more than a Week	3
		Delayed more than a Week	1
R2	Neatness	Documented, Clean neat & properly maintain.	5
		Documented properly maintain	3
		Documented not properly maintain	1
R3	Knowledge & Implementation	Correct implementation with Results and oral	5
		Implementation with some errors, oral	3
		Partial implementation, oral	1

***means obtained marks will be scaled to 15 for Experiments**

DATTA MEGHE COLLEGE OF ENGINEERING
DEPARTMENT OF COMPUTER ENGINEERING
ACADEMIC YEAR 2023-24 (TERM I)
SUBJECT: SOFTWARE ENGINEERING LAB (CSL501)
SEM: V DIV: A&B
RUBRICS FOR GRADING ASSIGNMENTS

Rubric Number	Rubric Title	Criteria	Marks* (out of 5)
R1	Documentation	On-time	2
		Delayed by not more than a Week	1
		Delayed more than a Week	0
R2	Knowledge & Concept	Clear understanding	2
		Partially understood	1
		Weak understanding	0
R3	Punctuality, Completion Time / Timeline	Correct Documentation	1
		Not documented properly	0

*means obtained marks will be scaled to 5 for Assignments

EXPERIMENT NO.:01

Date of Performance:

Date of Submission:

Aim: Application of at least two traditional process models.

Software Used: Ms Word

Theory: Software Processes is a coherent set of activities for specifying, designing, implementing and testing software systems. A software process model is an abstract representation of a process that presents a description of a process from some particular perspective. There are many different software processes but all involve:

- Specification – defining what the system should do;
- Design and implementation – defining the organization of the system and implementing the system;
- Validation – checking that it does what the customer wants;
- Evolution – changing the system in response to changing customer needs.

Types of Software Process Model

Software processes, methodologies and frameworks range from specific prescriptive steps that can be used directly by an organization in day-to-day work, to flexible frameworks that an organization uses to generate a custom set of steps tailored to the needs of a specific project or group. In some cases a “sponsor” or “maintenance” organization distributes an official set of documents that describe the process.

Software Process and Software Development Lifecycle Model

One of the basic notions of the software development process is SDLC models which stands for Software Development Life Cycle models. There are many development life cycle models that have been developed in order to achieve different required objectives. The models specify the various stages of the process and the order in which they are carried out. The most used, popular and important SDLC models are given below:

- Waterfall model
- V model
- Incremental model
- RAD model
- Agile model
- Iterative model
- Spiral model
- Prototype model

- Student can draw two traditional process models and write its application in details (1.Which process model used in Mini project. 2. Anyone other than mini project).

Conclusion:

Sign and Remark:

R1	R2	R3	Total Marks	Signature
(5)	(5)	(5)	(15)	

EXPERIMENT NO.:02

Date of Performance:

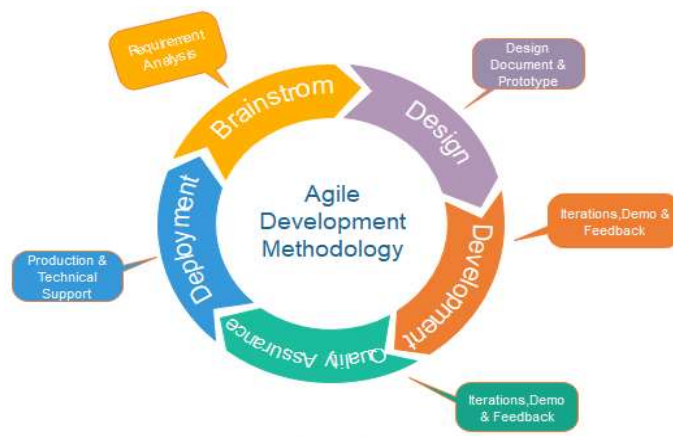
Date of Submission:

Aim: Application of the Agile process models.

Software Used: JIRA

Theory: The meaning of Agile is swift or versatile. “**Agile process model**” refers to a software development approach based on iterative development. Agile methods break tasks into smaller iterations, or parts do not directly involve long term planning. The project scope and requirements are laid down at the beginning of the development process. Plans regarding the number of iterations, the duration and the scope of each iteration are clearly defined in advance.

Each iteration is considered as a short time "frame" in the Agile process model, which typically lasts from one to four weeks. The division of the entire project into smaller parts helps to minimize the project risk and to reduce the overall project delivery time requirements. Each iteration involves a team working through a full software development life cycle including planning, requirements analysis, design, coding, and testing before a working product is demonstrated to the client.



Every iteration involves cross functional teams working simultaneously on various areas like

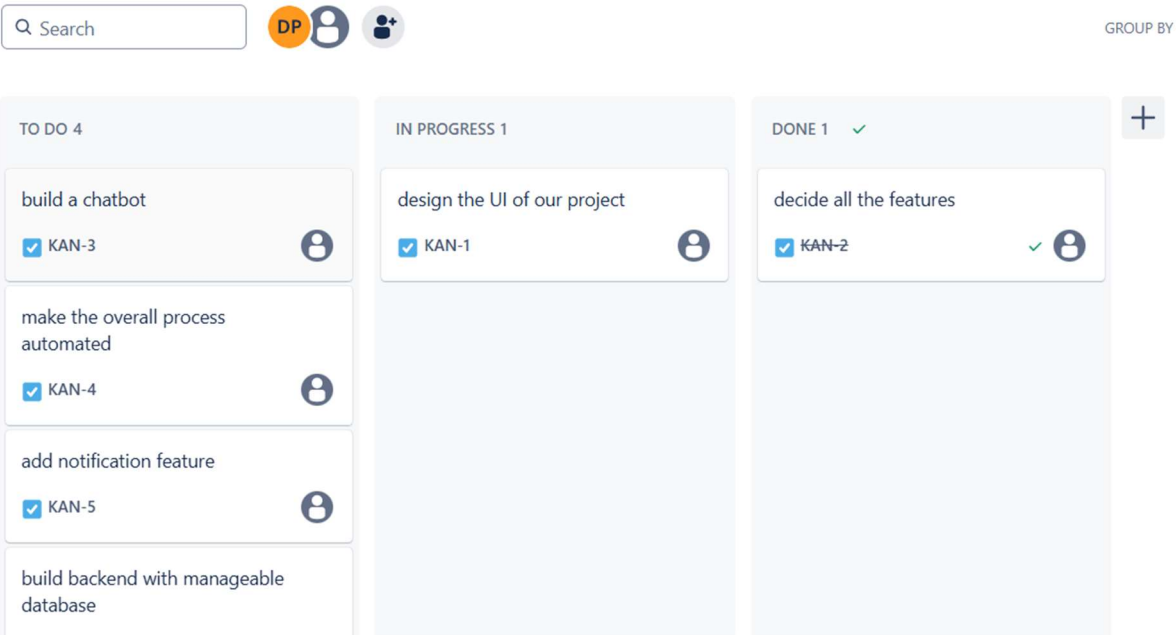
- Planning
- Requirements Analysis
- Design
- Coding
- Unit Testing and
- Acceptance Testing.

At the end of the iteration, a working product is displayed to the customer and important stakeholders.

Kanban methodology is based on the idea of continuous releases. Work is tracked using a kanban board that displays the statuses of work in columns and lanes. There are four important pillars to kanban to help teams ship products: continuous releases, WIP (work in progress) limits, the list of work, and columns or lanes. Here are some tools that come out-of-the-box in Jira Software's kanban template to help you run kanban with your team.

Projects / Automated Ironing services

KAN board



Conclusion:

Agile process models enable adaptive planning and iterative development, enhancing responsiveness to change and improving project outcomes through collaboration.

Sign and Remark:

R1	R2	R3	Total Marks	Signature
(5)	(5)	(5)	(15)	

EXPERIMENT NO.:03

Date of Performance:

Date of Submission:

Aim: Preparation of Software Requirement Specification (SRS) document in IEEE format

Software Used: Ms-Word

Theory:

The purpose of this document is to give a detailed description of the requirements for the software. It will illustrate the purpose and complete declaration for the development of system. It will also explain system constraints, interface and interactions with other external applications. This document is primarily intended to be proposed to a customer for its approval and a reference for developing the first version of the system for the development team.

Table of Contents

Introduction

1.1 Purpose

1.2 Scope

1.3 Definitions, acronyms, and abbreviations

1.4 References

2. Overall description

2.1 Product perspective

2.2 Product functions

2.3 User characteristics

2.4 Constraints

2.5 Assumptions and dependencies

2.6 Apportioning of requirements

3. Specific requirements

3.1.1 User interfaces

3.1.2 Hardware interfaces

- 3.1.3 Software interfaces
- 3.1.4 Communications interfaces
- 3.2 Functional requirements
 - 3.2.1 User Class 1
 - 3.2.2 User Class 2
 - 3.2.3 User Class 3
- 3.3 Performance requirements
- 3.4 Design constraints
- 3.5 Software system attributes
- 4. Prioritization and Release Plan
 - 4.1 Choice of prioritization method
- Appendix

Introduction

1.1 Purpose

This document describes the requirements for a mobile app that allows users to schedule ironing services when needed. The app will let users arrange for their clothes to be picked up, ironed, and delivered back. Users can also make payments and track their orders through the app.

1.2 Scope

The app will provide the following key features:

- Users can schedule ironing services.
- They can choose when their clothes will be picked up and delivered.
- They can pay for the services securely through the app.
- Users can check the status of their orders in real-time.

The application will be available on both iOS and Android.

1.3 Definitions , Acronyms , and Abbreviations

- **Ironing Service:** Pressing clothes to remove wrinkles.
- **Application:** The mobile application available on smartphones.
- **User:** Anyone using the app to request ironing services.
- **Payment Gateway:** A service that processes payments securely through the application.

1.4 References

ISO/IEC 9126: Software Engineering - Product Quality.

IEEE 830: IEEE Recommended Practice for Software Requirements Specifications.

2. Overall Description

2.1 Product Perspective

The mobile application will serve as a platform for scheduling and managing on-demand ironing services. It will provide a user-friendly interface where users can arrange garment pick-ups, track ironing orders, and receive their clothes once ironed. The app will streamline the process by offering service scheduling, pick-up and delivery options, secure payments, and real-time order tracking.

2.2 Product Functions

- **Service Scheduling:** Users can select times for garment pick-up and delivery.
- **Order Tracking:** The app allows users to track the progress of their ironing orders in real time.
- **Payments:** Users can securely pay for the services through the app.
- **Notifications:** Users will receive alerts about the status of their orders.
- **Service History:** Users can view past orders for reference.

2.3 User Characteristics

End Users: Individuals who need professional ironing services.

These users may have varying levels of technical expertise but require a simple and intuitive app interface to schedule services and monitor their orders.

2.4 Constraints

- **Device Compatibility:** The app will be designed for iOS and Android devices.
- **Service Availability:** The ironing services will depend on the availability of local providers.
- **Internet Connectivity:** Users will need internet access to schedule services, process payments, and track orders.
- **Payment Security:** The app must ensure secure handling of payments and personal data through encryption.

2.5 Assumptions and Dependencies

Availability of Service Providers: Local ironing service providers are available and integrated with the app.

- **Payment Gateway:** Secure payment systems are in place to process transactions.
- **Internet Access:** Users need internet connectivity to interact with the app.

2.6 Apportioning of Requirements

The initial development will focus on implementing the core functionalities of the app, including: Scheduling pick-up and delivery for ironing services. Processing secure Payment.

3. Specific Requirements

3.1.1 User Interfaces

- **Web Interface:** A simple, intuitive interface allowing users to:
 - Select clothing items for ironing.
 - View pricing and offers.
 - Track the status of their ironing process (pickup, ironing, delivery).
 - Provide feedback or request additional services.

Mobile Interface: A mobile-friendly version of the web application with the same functionality, optimized for smaller screens.

3.1.2 Hardware Interfaces

Server Requirements: The application will run on a web server capable of handling user requests and updates efficiently. The mobile version will require integration with mobile-specific APIs for location tracking.

3.1.3 Software Interfaces

- **Payment Gateway:** Integration with payment services (like Stripe or Razorpay) for secure transactions.
- **SMS API:** Integration with SMS services for sending updates to users.

3.1.4 Communications Interfaces

Network: The application will require internet access for cloud-based data processing, tracking, and communication with users.

3.2 Functional Requirements

3.2.1 User Class 1: End Users

- **Clothing Item Selection:** Users must be able to choose clothing items (shirts, pants, etc.) from a predefined list.
- **Price Calculation:** The system will display pricing based on the selected items.
- **Order Status Tracking:** Users can view the status of their ironing order (pickup, ironing, delivery).
- **Service Request:** Users should be able to request pickup and delivery times based on their convenience.

- **Order Update Notifications:** Users must receive updates via email or SMS on the progress of their ironing order.
- **Payment:** Users should be able to make secure payments for their services.

3.2.2 User Class 2: Admins

- **Order Management:** Admins must be able to view, manage, and update orders.
- **Customer Database:** Admins will manage customer information and past orders.
- **Driver Assignments:** Admins will assign drivers for pickups and deliveries.

3.3 Performance Requirements

System Responsiveness: The web and mobile application should respond within 2 seconds to user actions.

Order Processing Time: The system should reflect updated statuses (pickup, ironing, delivery) in real-time, within 30 seconds of an action.

3.4 Design Constraints

Compatibility: The application must be compatible with modern web browsers (e.g., Chrome, Firefox, Edge) and mobile operating systems (Android, iOS).

Security: Ensure secure handling of user data, including encrypted communication for sensitive information such as payment details.

3.5 Software System Attributes

Usability: The application should have an easy-to-navigate interface for users, with minimal clicks required to place an order.

Reliability: The system should be available 99% of the time and have minimal downtime.

Security: User data, including order history and payment details, must be protected with encryption and secure protocols.

Scalability: The system should be scalable to handle up to 10,000 concurrent users in metropolitan cities.

4. Prioritization and Release Plan

4.1 Choice of Prioritization Method

MoSCoW Method:

- Must Have: Catalog , Tracking , Payment , Alerts.
- Should Have: Mobile app integration with location tracking for drivers. Special offers and discounts for loyal customers.
- Could Have: Advanced customer preferences and feedback system, Custom service requests for bulk orders.
- Won't Have: Advanced features like subscription-based services in the initial release.

Appendix

A. Glossary: Terms such as "Order Status," "Clothing Items," "Notifications," etc.

B. Change History: Record of changes made to this SRS document.

C. Mockups: (Optional) Visual representations of the user interface for clothing item selection, order status, and payment

Conclusion:

The SRS for our mini project has been successfully prepared, detailing its key formalities.

Sign and Remark:

R1	R2	R3	Total Marks	Signature
(5)	(5)	(5)	(15)	

EXPERIMENT NO.:04

Date of Performance:

Date of Submission:

Aim: Structured Data Flow Analysis

Software Used: Star UML

Theory: Structured Analysis is a development method that allows the analyst to understand the system and its activities in a logical way.

It is a systematic approach, which uses graphical tools that analyze and refine the objectives of an existing system and develop a new system specification which can be easily understandable by user.

It has following attributes –

- It is graphic which specifies the presentation of application.
- It divides the processes so that it gives a clear picture of system flow.
- It is logical rather than physical i.e., the elements of system do not depend on vendor or hardware.
- It is an approach that works from high-level overviews to lower-level details.

Structured Analysis Tools

During Structured Analysis, various tools and techniques are used for system development. They are –

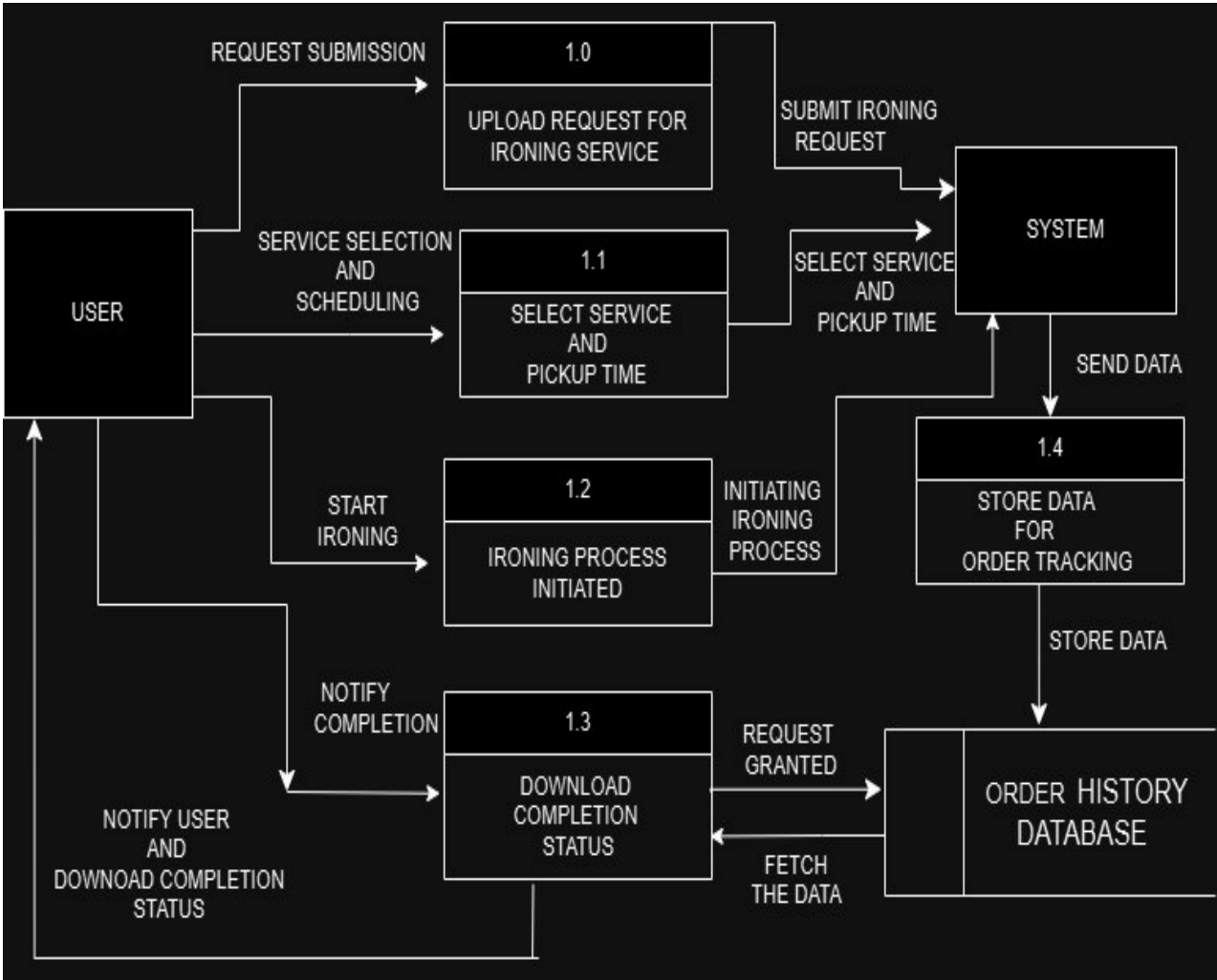
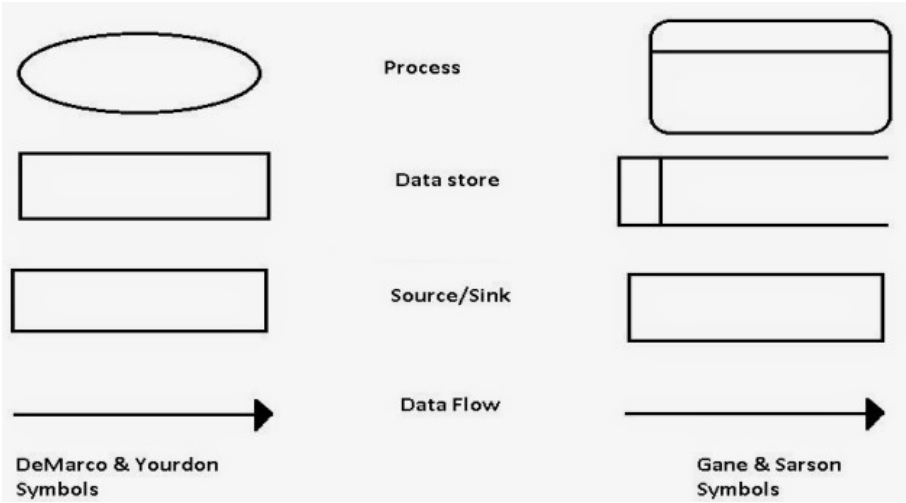
- Data Flow Diagrams
- Data Dictionary
- Decision Trees
- Decision Tables
- Structured English
- Pseudo code

Data Flow Diagrams (DFD)

It is a technique developed by Larry Constantine to express the requirements of system in a graphical form.

- It shows the flow of data between various functions of system and specifies how the current system is implemented.
- It is an initial stage of design phase that functionally divides the requirement specifications down to the lowest level of detail.
- Its graphical nature makes it a good communication tool between user and analyst or analyst and system designer.
- It gives an overview of what data a system processes, what transformations are performed, what data are stored, what results are produced and where they flow.

Symbols used in DFD



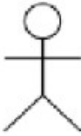
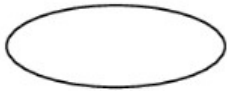
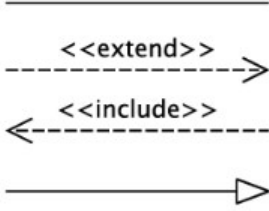
Used Case Diagram

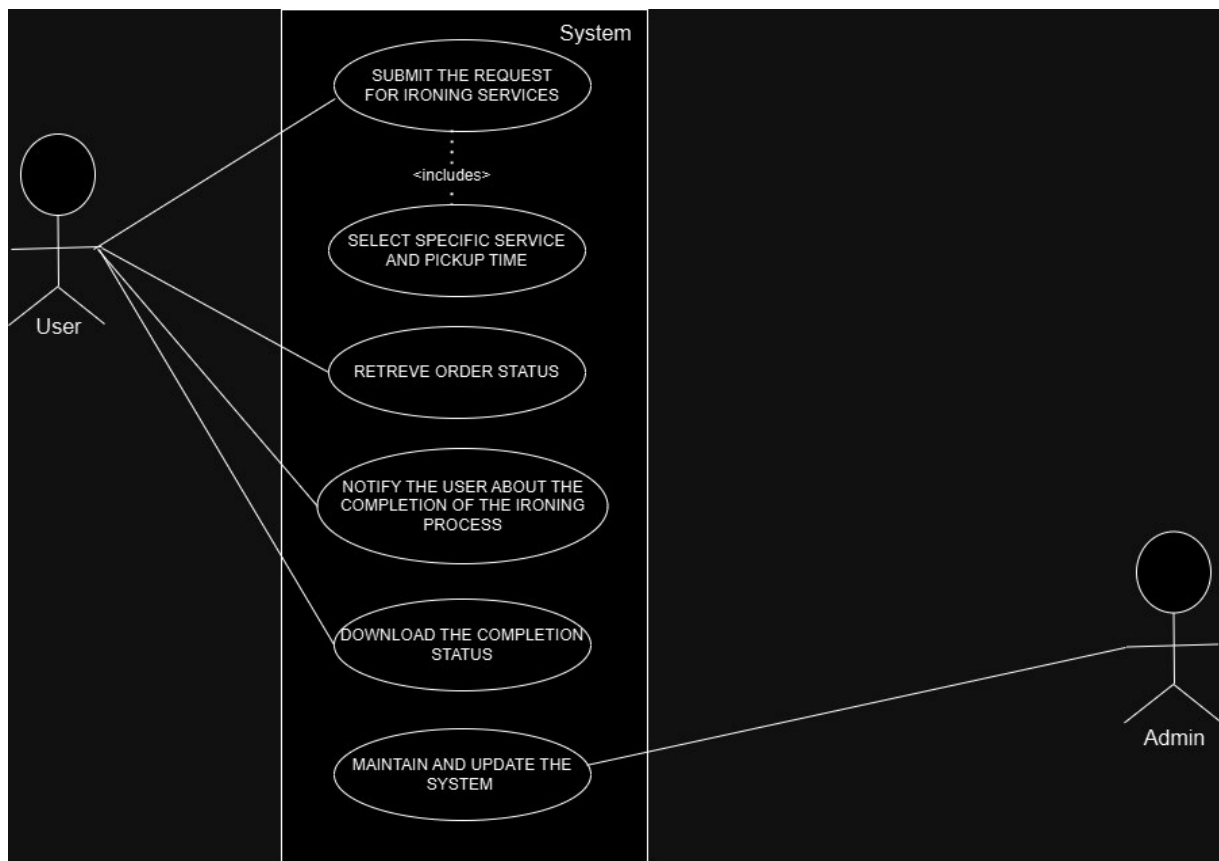
In the Unified Modeling Language (UML), a use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system. To build one, you'll use a set of specialized symbols and connectors. An effective use case diagram can help your team discuss and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

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

UML use case diagrams are ideal for:

- Representing the goals of system-user interactions
- Defining and organizing functional requirements in a system
- Specifying the context and requirements of a system
- Modeling the basic flow of events in a use case

Symbol	Reference Name
	Actor
	Use case
	Relationship



Conclusion:

Hence we analyzed successfully the data flow.

Sign and Remark:

R1	R2	R3	Total Marks	Signature
(5)	(5)	(5)	(15)	

EXPERIMENT NO.:05

Date of Performance:

Date of Submission:

Aim: Use of Metrics to estimate the cost

Software Used: Ms-Word

Software Cost Estimation

For any new software project, it is necessary to know how much it will cost to develop and how much development time will it take. These estimates are needed before development is initiated, but how is this done? Several estimation procedures have been developed and are having the following attributes in common.

1. Project scope must be established in advanced.
2. Software metrics are used as a support from which evaluation is made.
3. The project is broken into small PCs which are estimated individually.
To achieve true cost & schedule estimate, several option arise.
4. Delay estimation
5. Used symbol decomposition techniques to generate project cost and schedule estimates.
6. Acquire one or more automated estimation tools.

Uses of Cost Estimation

1. During the planning stage, one needs to choose how many engineers are required for the project and to develop a schedule.
2. In monitoring the project's progress, one needs to access whether the project is progressing according to the procedure and takes corrective action, if necessary.
3. Estimation is the process of finding an estimate, or approximation, which is a value that can be used for some purpose even if input data may be incomplete, uncertain, or unstable.
4. Estimation determines how much money, effort, resources, and time it will take to build a specific system or product.

Loc-based Cost Estimation

The LOC (Line of Code) is a product size metric in software engineering. Here, the number of lines in the code are counted and based on the number of lines the cost is calculated.

LOC-based Estimation

- Different languages lead to different lengths of code
- It is not clear how to count lines of code
- A report, screen, or GUI generator, can generate thousands of lines of code in minutes
- Depending on the application, the complexity of code is different

Function Points: FP

Function Points is used in 2 contexts:

- **Past:** To develop **metrics** from historical data
- **Future:** Use of available metrics to size the s/w of a new project

FP-based Estimation

- Based on FP metric for the size of a product
- Based on the number of inputs (Inp), outputs (Out), inquiries (Inq), master files (Maf), interfaces (Inf)
- Classify each component of the product (Inp, Out, Inq, Maf, Inf) as simple, average, or complex (next slide)

LOC-based Cost Estimation:

Component	Estimated LOC
Service Request Upload & Processing	1000 LOC
Service Scheduling System	1000 LOC
User Interface (UI)	800 LOC
Database Management	600 LOC
API Integration	400 LOC
Error Handling and Logging	300 LOC
Security Features	400 LOC

Total Estimated LOC = 500 + 1000 + 800 + 600 + 400 + 300 + 400 = 4000 LOC

Cost Per LOC = 8000/320 = Rs. 25/ LOC

Total Estimated Project Cost = 4000 LOC X 25 Rs./LOC = Rs. 100,000

Estimated Efforts (Person Months) = $100,000/8000 = 12.5$ person – months

FP-Based Estimation:

Parameter	Estimated Count	Weight	Total
No. Of User Input	2	4	8
No. Of User Output	2	5	10
No. Of User Enquire	1	4	4
No. Of User Files	2	10	20
No. Of User External Interfaces	8	10	80

Total Count = $8 + 10 + 4 + 20 + 80 = 122$

$\Sigma Fi = 42$

Function Point = Total Count $\times [0.65 + 0.01 \times \Sigma Fi]$

Function Point = $122 \times [0.65 + 0.42]$

Function Point = $122 \times 1.07 = 130.54$

Conclusion:

This FP estimation provides a metric to assess the cost, effort, and complexity of your automated ironing service system.

Sign and Remark:

R1	R2	R3	Total Marks	Signature
(5)	(5)	(5)	(15)	

EXPERIMENT NO.:06

Date of Performance:

Date of Submission:

Aim: Scheduling and tracking of the project

Software Used: Gantt Project

Theory:- Project Scheduling

Project-task scheduling is a significant project planning activity. It comprises deciding which functions would be taken up when. To schedule the project plan, a software project manager wants to do the following:

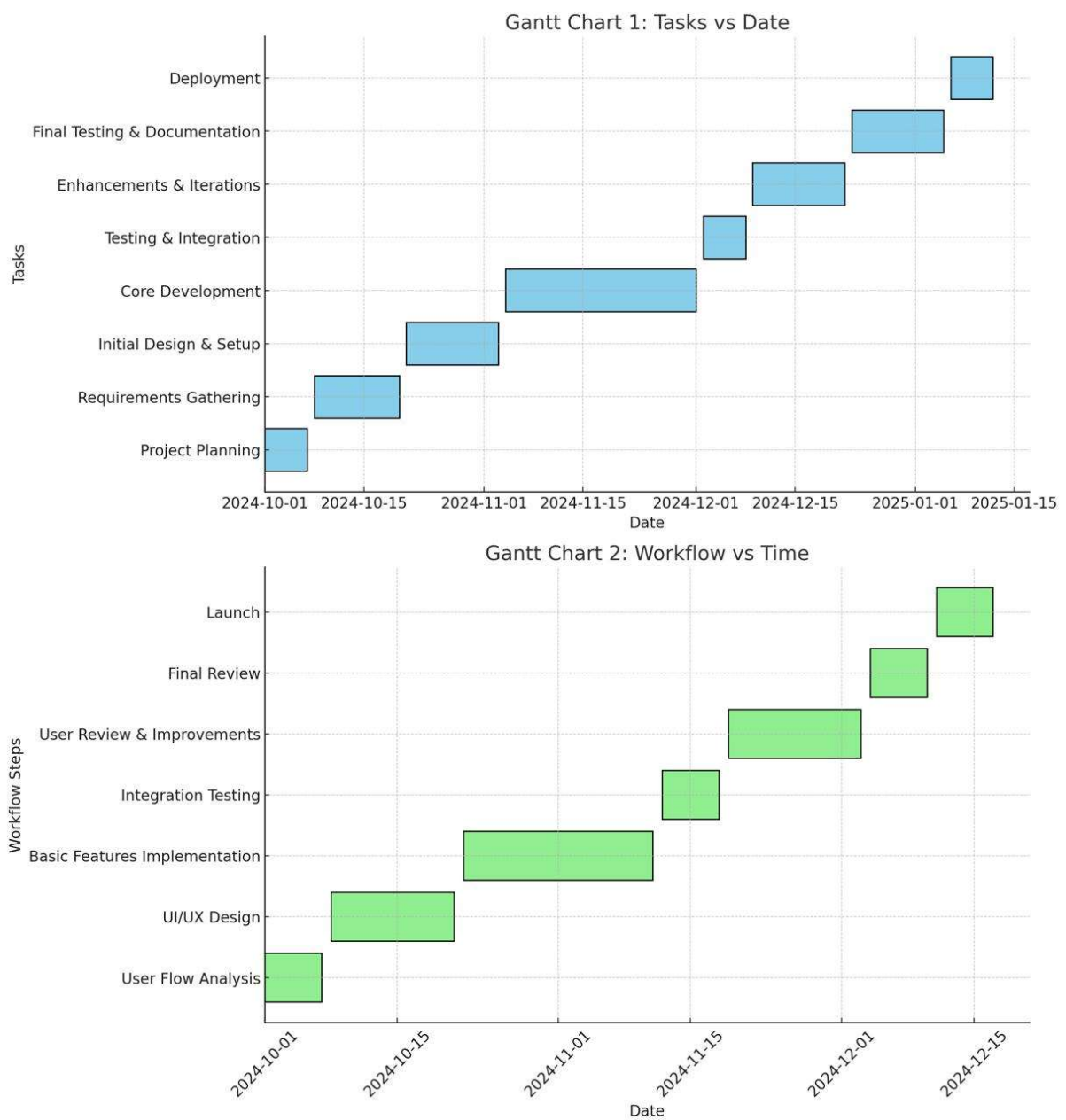
1. Identify all the functions required to complete the project.
2. Break down large functions into small activities.
3. Determine the dependency among various activities.
4. Establish the most likely size for the time duration required to complete the activities.
5. Allocate resources to activities.
6. Plan the beginning and ending dates for different activities.
7. Determine the critical path. A critical way is the group of activities that decide the duration of the project.

The first method in scheduling a software plan involves identifying all the functions required to complete the project. A good judgment of the intricacies of the project and the development process helps the supervisor to identify the critical role of the project effectively. Next, the large functions are broken down into a valid set of small activities which would be assigned to various engineers. The work breakdown structure formalism supports the manager to breakdown the function systematically after the project manager has broken down the purpose and constructs the work breakdown structure; he has to find the dependency among the activities. Dependency among the various activities determines the order in which the various events would be carried out. If an activity A necessary the results of another activity B, then activity A must be scheduled after activity B. In general, the function dependencies describe a partial ordering among functions, i.e., each service may precede a subset of other functions, but some functions might not have any precedence ordering describe between them (called concurrent function). The dependency among the activities is defined in the pattern of an activity network.

Once the activity network representation has been processed out, resources are allocated to every activity. Resource allocation is usually done using a Gantt chart. After resource allocation

is completed, a PERT chart representation is developed. The PERT chart representation is useful for program monitoring and control. For task scheduling, the project plan needs to decompose the project functions into a set of activities. The time frame when every activity is to be performed is to be determined. The end of every action is called a milestone. The project manager tracks the function of a project by audit the timely completion of the milestones. If he examines that the milestones start getting delayed, then he has to handle the activities carefully so that the complete deadline can still be met.

Gantt Chart :



Conclusion:

Thus, we have scheduled and tracked our project using Gantt chart.

Sign and Remark:

R1	R2	R3	Total Marks	Signature
(5)	(5)	(5)	(15)	

EXPERIMENT NO.:07

Date of Performance:

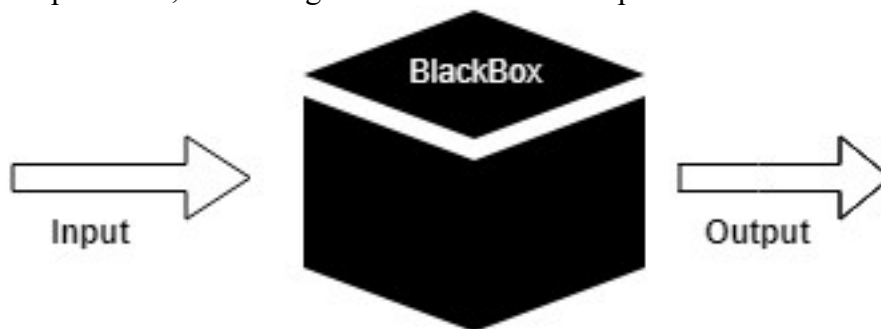
Date of Submission:

Aim: Write test cases for black box testing

Software Used: Selenium/GitHub/Jira

Theory:- Black Box Testing is a software testing method in which the functionalities of software applications are tested without having knowledge of internal code structure, implementation details and internal paths. Black Box Testing mainly focuses on input and output of software applications and it is entirely based on software requirements and specifications. It is also known as Behavioral Testing.

In this method, tester selects a function and gives input value to examine its functionality, and checks whether the function is giving expected output or not. If the function produces correct output, then it is passed in testing, otherwise failed. The test team reports the result to the development team and then tests the next function. After completing testing of all functions if there are severe problems, then it is given back to the development team for correction.



Generic steps of black box testing

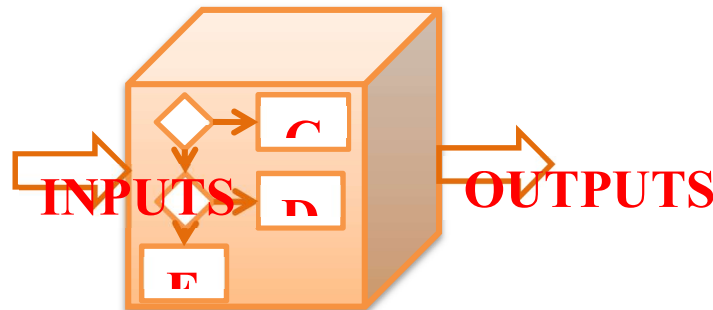
- The black box test is based on the specification of requirements, so it is examined in the beginning.
- In the second step, the tester creates a positive test scenario and an adverse test scenario by selecting valid and invalid input values to check that the software is processing them correctly or incorrectly.
- In the third step, the tester develops various test cases such as decision table, all pairs test, equivalent division, error estimation, cause-effect graph, etc.
- The fourth phase includes the execution of all test cases.
- In the fifth step, the tester compares the expected output against the actual output.
- In the sixth and final step, if there is any flaw in the software, then it is cured and tested again.

White Box Testing

White Box Testing is software testing technique in which internal structure, design and coding of software are tested to verify flow of input-output and to improve design, usability and

security. In white box testing, code is visible to testers so it is also called Clear box testing, Open box testing, Transparent box testing, Code-based testing and Glass box testing.

The developer will test every line of the code of the program. The developers perform the White-box testing and then send the application or the software to the testing team, where they will perform the black box testing and verify the application along with the requirements and identify the bugs and sends it to the developer.



Test procedure

The test procedure of black box testing is a kind of process in which the tester has specific knowledge about the software's work, and it develops test cases to check the accuracy of the software's functionality.

It does not require programming knowledge of the software. All test cases are designed by considering the input and output of a particular function. A tester knows about the definite output of a particular input, but not about how the result is arising. There are various techniques used in black box testing for testing like decision table technique, boundary value analysis technique, state transition, All-pair testing, cause-effect graph technique, equivalence partitioning technique, error guessing technique, use case technique and user story technique.

Test cases

Test cases are created considering the specification of the requirements. These test cases are generally created from working descriptions of the software including requirements, design parameters, and other specifications. For the testing, the test designer selects both positive test scenario by taking valid input values and adverse test scenario by taking invalid input values to determine the correct output. Test cases are mainly designed for functional testing but can also be used for non-functional testing. Test cases are designed by the testing team; there is not any involvement of the development team of software.

Techniques Used in Black Box Testing

Decision Table Technique	Decision Table Technique is a systematic approach where various input combinations and their respective system behavior are captured in a tabular form. It is appropriate for the functions that have a logical relationship between two and more than two inputs.
Boundary Value Technique	Boundary Value Technique is used to test boundary values, boundary values are those that contain the upper and lower limit of a variable. It tests, while entering boundary value whether the software is producing correct output or not.

State Transition Technique	State Transition Technique is used to capture the behavior of the software application when different input values are given to the same function. This applies to those types of applications that provide the specific number of attempts to access the application.
All-pair Testing Technique	All-pair testing Technique is used to test all the possible discrete combinations of values. This combinational method is used for testing the application that uses checkbox input, radio button input, list box, text box, etc.
Cause-Effect Technique	Cause-Effect Technique underlines the relationship between a given result and all the factors affecting the result. It is based on a collection of requirements.
Equivalence Partitioning Technique	Equivalence partitioning is a technique of software testing in which input data divided into partitions of valid and invalid values, and it is mandatory that all partitions must exhibit the same behavior.
Error Guessing Technique	Error guessing is a technique in which there is no specific method for identifying the error. It is based on the experience of the test analyst, where the tester uses the experience to guess the problematic areas of the software.
Use Case Technique	Use case Technique used to identify the test cases from the beginning to the end of the system as per the usage of the system. By using this technique, the test team creates a test scenario that can exercise the entire software based on the functionality of each function from start to end.

Conclusion:

Link for Download software :- GitHub:- <https://github.com/features/actions>
 Jira:- <https://www.atlassian.com/try/cloud/signup?bundle=jira-software&edition=free>
 Selenium:- <https://www.selenium.dev/downloads>

Sign and Remark:

R1	R2	R3	Total Marks	Signature
(5)	(5)	(5)	(15)	

EXPERIMENT NO.:08

Date of Performance:

Date of Submission:

Aim: Write test cases for white box testing

Software Used: : Selenium/GitHub/Jira

Theory:-

White Box Testing is software testing technique in which internal structure, design and coding of software are tested to verify flow of input-output and to improve design, usability and security. In white box testing, code is visible to testers so it is also called Clear box testing, Open box testing, Transparent box testing, Code-based testing and Glass box testing.

White box testing which also known as glass box is **testing, structural testing, clear box testing, open box testing and transparent box testing**. It tests internal coding and infrastructure of a software focus on checking of predefined inputs against expected and desired outputs. It is based on inner workings of an application and revolves around internal structure testing. In this type of testing programming skills are required to design test cases. The primary goal of white box testing is to focus on the flow of inputs and outputs through the software and strengthening the security of the software.

The term 'white box' is used because of the internal perspective of the system. The clear box or white box or transparent box name denote the ability to see through the software's outer shell into its inner workings.

Developers do white box testing. In this, the developer will test every line of the code of the program. The developers perform the White-box testing and then send the application or the software to the testing team, where they will perform the black box testing and verify the application along with the requirements and identify the bugs and sends it to the developer.

The developer fixes the bugs and does one round of white box testing and sends it to the testing team. Here, fixing the bugs implies that the bug is deleted, and the particular feature is working fine on the application.

Here, the test engineers will not include in fixing the defects for the following reasons:

- Fixing the bug might interrupt the other features. Therefore, the test engineer should always find the bugs, and developers should still be doing the bug fixes.
- If the test engineers spend most of the time fixing the defects, then they may be unable to find the other bugs in the application.

Techniques Used in White Box Testing

Data Flow Testing	Data flow testing is a group of testing strategies that examines the control flow of programs in order to explore the sequence of variables according to the sequence of events.
Control Flow Testing	Control flow testing determines the execution order of statements or instructions of the program through a control structure. The control structure of a program is used to develop a test case for the program. In this technique, a particular part of a large program is selected by the tester to set the testing path. Test cases represented by the control graph of the program.
Branch Testing	Branch coverage technique is used to cover all branches of the control flow graph. It covers all the possible outcomes (true and false) of each condition of decision point at least once.
Statement Testing	Statement coverage technique is used to design white box test cases. This technique involves execution of all statements of the source code at least once. It is used to calculate the total number of executed statements in the source code, out of total statements present in the source code.
Decision Testing	This technique reports true and false outcomes of Boolean expressions. Whenever there is a possibility of two or more outcomes from the statements like do while statement, if statement and case statement (Control flow statements), it is considered as decision point because there are two outcomes either true or false.

Let us consider the following code :

```
INPUT A & B  
  
C = A + B  
  
IF C>100  
  
PRINT "ITS DONE"
```

Now in the first, line, we assign the value of A and B. Let us suppose A = 60 and B = 50. Moving on to the second line, now C is assigned a value of A+B, here A = 60 and B = 50, hence C = 110. Moving on to the third line, we will check if C > 100, here the condition is true and hence we should get our result as ITS DONE

Conclusion:

Link for Download software https: Link for Download software :- GitHub:-

<https://github.com/features/actions>

Jira:- <https://www.atlassian.com/try/cloud/signup?bundle=jira-software&edition=free>

Selenium:- <https://www.selenium.dev/downloads>

Sign and Remark:

R1	R2	R3	Total Marks	Signature
(5)	(5)	(5)	(15)	

EXPERIMENT NO.:09

Date of Performance:

Date of Submission:

Aim: Preparation of Risk Mitigation, Monitoring and Management plan (RMMM.)

Software Used: Ms Word

Theory:-

RMMM Plan :

A risk management technique is usually seen in the software Project plan. This can be divided into Risk Mitigation, Monitoring, and Management Plan (RMMM). In this plan, all works are done as part of risk analysis. As part of the overall project plan project manager generally uses this RMMM plan.

In some software teams, risk is documented with the help of a Risk Information Sheet (RIS). This RIS is controlled by using a database system for easier management of information i.e creation, priority ordering, searching, and other analysis. After documentation of RMMM and start of a project, risk mitigation and monitoring steps will start.

Risk Mitigation :

It is an activity used to avoid problems (Risk Avoidance).

Steps for mitigating the risks as follows.

1. Finding out the risk.
2. Removing causes that are the reason for risk creation.
3. Controlling the corresponding documents from time to time.
4. Conducting timely reviews to speed up the work.

Risk Monitoring :

It is an activity used for project tracking.

It has the following primary objectives as follows.

To check if predicted risks occur or not.

1. To ensure proper application of risk aversion steps defined for risk.
2. To collect data for future risk analysis.
3. To allocate what problems are caused by which risks throughout the project.

Risk Management and planning :

It assumes that the mitigation activity failed and the risk is a reality. This task is done by Project manager when risk becomes reality and causes severe problems. If the project manager effectively uses project mitigation to remove risks successfully then it is easier to manage the risks. This shows that the response that will be taken for each risk by a manager. The main objective of the risk management plan is the risk register. This risk register describes and focuses on the predicted threats to a software project.

Steps for Risk Management

1. Identify possible risks and recognize what can go wrong

2. Analyse each risk to estimate the probability that it will occur and the impact (i.e., damage) that it will do if it does occur
3. Rank the risks by probability and impact. Impact may be negligible, marginal, critical, and catastrophic.
4. Develop a contingency plan to manage those risks having high probability and high impact

Risk Table

Risks ID	Risks	Category	Probability	Impact

Impact Values: 1 – Catastrophic 2 – Critical 3 – Marginal 4 – Negligible

Conclusion:

Sign and Remark:

R1	R2	R3	Total Marks	Signature
(5)	(5)	(5)	(15)	

EXPERIMENT NO.:10

Date of Performance:

Date of Submission:

Aim: Version control of the project

Software Used: GitHub

Theory:-

Version control systems are a category of software tools that helps in recording changes made to files by keeping a track of modifications done to the code.

As we know that a software product is developed in collaboration by a group of developers they might be located at different locations and each one of them contributes in some specific kind of functionality/features. So in order to contribute to the product, they made modifications in the source code(either by adding or removing). A version control system is a kind of software that helps the developer team to efficiently communicate and manage (track) all the changes that have been made to the source code along with the information like who made and what change has been made. A separate branch is created for every contributor who made the changes and the changes aren't merged into the original source code unless all are analyzed as soon as the changes are green signaled they merged to the main source code. It not only keeps source code organized but also improves productivity by making the development process smooth.

Benefits of the version control system:

- a) Enhances the project development speed by providing efficient collaboration,
- b) Leverages the productivity, expedite product delivery, and skills of the employees through better communication and assistance,
- c) Reduce possibilities of errors and conflicts meanwhile project development through traceability to every small change,
- d) Employees or contributor of the project can contribute from anywhere irrespective of the different geographical locations through this VCS,
- e) For each different contributor of the project a different working copy is maintained and not merged to the main file unless the working copy is validated. A most popular example is **Git, Helix core, Microsoft TFS**,
- f) Helps in recovery in case of any disaster or contingent situation,
- g) Informs us about Who, What, When, Why changes have been made.

Use of Version Control System:

- **A repository:** It can be thought of as a database of changes. It contains all the edits and historical versions (snapshots) of the project.
- **Copy of Work (sometimes called as checkout):** It is the personal copy of all the files in a project. You can edit to this copy, without affecting the work of others and you can finally commit your changes to a repository when you are done making your changes.

Conclusion:

Sign and Remark:

R1	R2	R3	Total Marks	Signature
(5)	(5)	(5)	(15)	

EXPERIMENT NO.:11

Date of Performance:

Date of Submission:

Aim: The context diagram of mess management

Lab Objectives:

- 1 To solve real life problems by applying software engineering principles
- 2 To impart state-of-the-art knowledge on Software Engineering
- **Lab Outcomes:**
- On successful completion of laboratory experiments, learners will be able to :
 - 1. Develop architectural models for the selected case study.
 - 2. Use computer-aided software engineering (CASE) tools.

Software Used: Star UML

Theory: Structured Analysis is a development method that allows the analyst to understand the system and its activities in a logical way.

It is a systematic approach, which uses graphical tools that analyze and refine the objectives of an existing system and develop a new system specification which can be easily understandable by user.

It has following attributes –

- It is graphic which specifies the presentation of application.
- It divides the processes so that it gives a clear picture of system flow.
- It is logical rather than physical i.e., the elements of system do not depend on vendor or hardware.
- It is an approach that works from high-level overviews to lower-level details.

Structured Analysis Tools

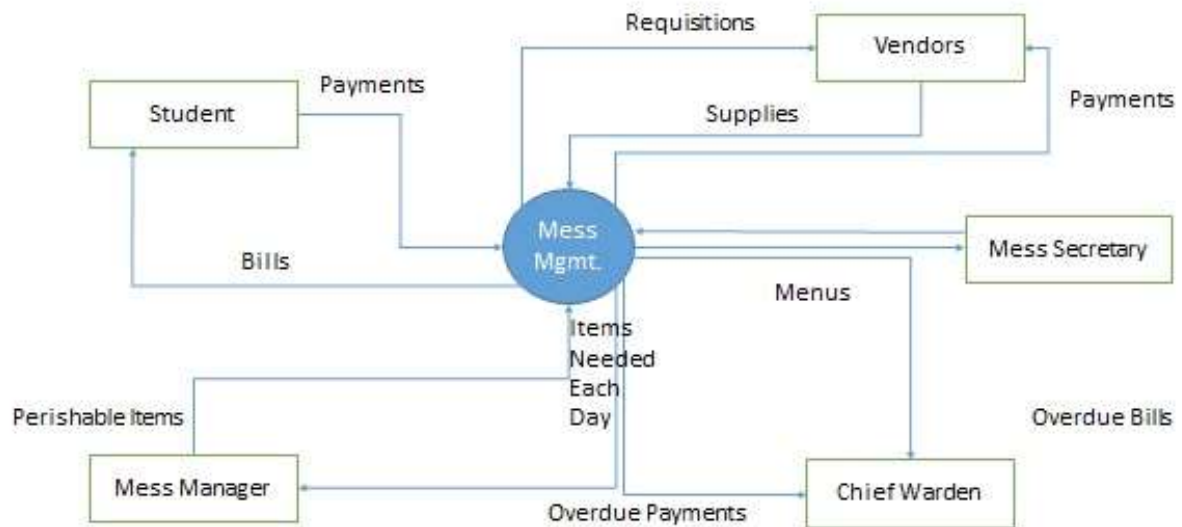
During Structured Analysis, various tools and techniques are used for system development. They are –

- Data Flow Diagrams
- Data Dictionary
- Decision Trees
- Decision Tables
- Structured English
- Pseudo code

Context Diagram

A context diagram helps in understanding the entire system by one DFD which gives the overview of a system. It starts with mentioning major processes with little details and then goes onto giving more details of the processes with the top-down approach.

The context diagram of mess management is shown below.



Used Case Diagram

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

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

UML use case diagrams are ideal for:

- Representing the goals of system-user interactions
- Defining and organizing functional requirements in a system
- Specifying the context and requirements of a system
- Modeling the basic flow of events in a use case

Conclusion:

Link to download software: <https://staruml.io/download>

Sign and Remark:

R1	R2	R3	Total Marks	Signature
(5)	(5)	(5)	(15)	

EXPERIMENT NO.:12

Date of Performance:

Date of Submission:

Aim: Activity Diagram for Online Shopping System

THEORY: An activity diagram shows the flow from activity to activity .An activity is an ongoing non atomic execution within a state machine .Activities ultimately results in some action, which is made up of executable atomic computations. We can use these diagrams to model the dynamic aspects of a system. Activity diagram is basically a flow chart to represent the flow form one activity to another . The activity can be described as an operation of the system. So the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent. Activity diagrams deals with all type of flow by using elements like fork, join etc.

Fork

A fork represents the splitting of a single flow of control into two or more concurrent Flow of control. A fork may have one incoming transition and two or more outgoing transitions, each of which represents an independent flow of control. Below fork the activities associated with each of these path continues in parallel.

Join

A join represents the synchronization of two or more concurrent flows of control. A join may have two or more incoming transition and one outgoing transition. Above the join the activities associated with each of these paths continues in parallel.

Branching

A branch specifies alternate paths takes based on some Boolean expression Branch is represented by diamond Branch may have one incoming transition and two or more outgoing one on each outgoing transition, you place a Boolean expression shouldn't overlap but they should cover all possibilities.

Swim lane:

Swim lanes are useful when we model workflows of business processes to partition the activity states on an activity diagram into groups. Each group representing the business organization responsible for those activities, these groups are called Swim lanes .

Procedure:-

Step1: First initial state is created.

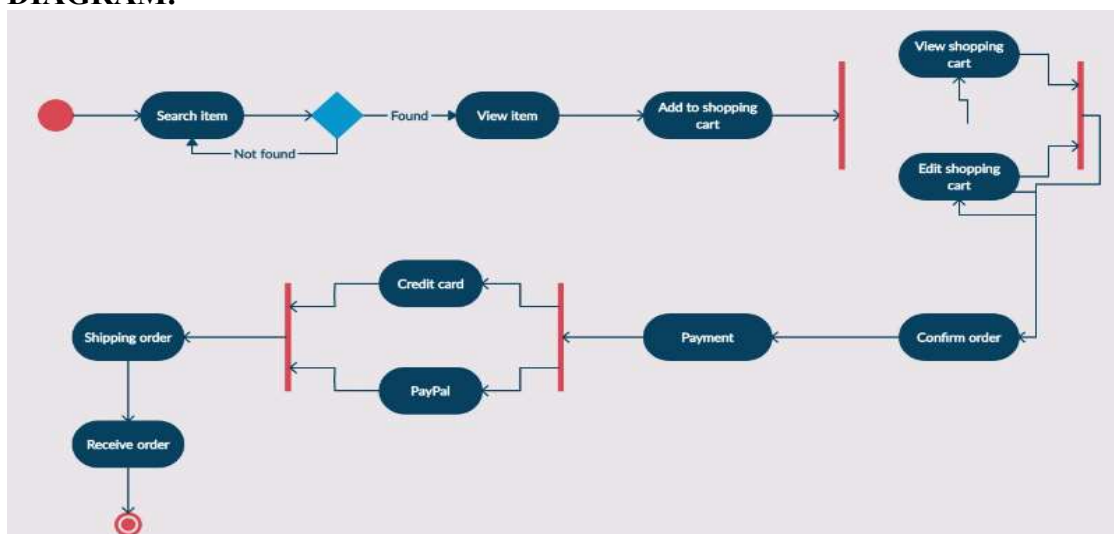
Step2: After that it goes to the action state insert card.

Step3: Next it undergoes transition to the state enter pin

Step4: In this way it undergoes transitions to the various states.

Step5: Use forking and joining wherever necessary.

DIAGRAM:



Conclusion:

Link to download software: <https://staruml.io/download>

Sign and Remark:

R1	R2	R3	Total Marks	Signature
(5)	(5)	(5)	(15)	