AN

INTERNSHIP REPORT ON

The Urban Services

Submitted

by

Mr. Meet Rana

(Enrollment Number: 221433142029)

Guided By:

Prof. Mohit Patel

An

Internship Report

Submitted to

Gujarat Technological University

In fulfillment for the award of degree of Bachelor of Engineering

in

CSE(AIML)

ACADEMIC YEAR – 2025

NEW L. J. INSTITUTE OF ENGINEERING AND TECHNOLOGY

Pakwan, Behind Rajpath Club Gate to Sindhu Bhavan Road,
Sarkhej - Gandhinagar Hwy, AEC Char Rasta, Ahmedabad, Gujarat 380054

NEW L. J. INSTITUTE OF ENGINEERING AND TECHNOLOGY

CERTIFICATE

This is to certify that the Internship report submitted along with the project entitled Internship in **Grownited Pvt Ltd** has been Completed by **Meet Rana** under my guidance in complete fulfilment for the Bachelor of Engineering in **CSE(AIML)** Branch, 8th Semester of **Gujarat Technological University**, Ahmedabad during the academic year 2025.

Date:

Place: NEW LJIET, Ahmedabad.

Signature and Name of Guide

Prof. Mohit Patel
Assistant Professor (CSE-AIML),
CSE(AIML)/IT Department,
NEW LJIET (143), Ahmedabad.

Signature and Name of H.O.D.

Dr. Gayatri Pandi.
Associate Professor, (CSE(AIML)/IT),
CSE(AIML)/IT Department,
NEW LJIET (143), Ahmedabad.

Signature and Name of Principal

Dr. Gayatri Pandi
NEW LJIET (143), Ahmedabad

Seal of Institute

GUJARAT TECHNOLOGICAL UNIVERSITY

CERTIFICATE FOR COMPLETION OF ALL ACTIVITIES AT ONLINE PROJECT PORTAL

B.E. SEMESTER VIII, ACADEMIC YEAR 2024-2025

Date of certificate generation : 17 April 2025 (11:25:01)

This is to certify that, ***Rana Meet*** (Enrolment Number - 221433142029) working on project entitled with ***Urban Service*** from ***CSE - ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING*** department of ***New L.J Institute of Engineering and Technology*** had submitted following details at online project portal.

Internship Project Report	Completed

Name of Student : Rana Meet

Name of Guide : Mr. MILAN JATINBHAI BHADALIYA
Mr. MOHIT BIPINBHAI PATEL

Signature of Student : _____

***Signature of Guide :** _____

Disclaimer :
This is a computer generated copy and does not indicate that your data has been evaluated. This is the receipt that CTU has received a copy of the data that you have uploaded and submitted as your project work.

*Guide has to sign the certificate, Only if all above activities has been Completed.

GROWNITED

Date: 14/04/2025

TO WHOMSOEVER IT MAY CONCERN

This is to certify that **Meet Rana** has successfully completed an internship in our organization as an intern in PYTHON for the duration of 3 Months.

He/She has worked on a project title **The Urban Services**.

During this tenure, we found the candidate to be hardworking, conscientious, and responsible. The feedback on his/her participation has always been positive and we wish him/her all the best.

For, Grownited Private Limited.

Rahul Kirpekar
(Authorised Signature)

301-305, 3rd Floor, Surbhi Complex, Nr. Municipal Market, Chimanlal Girdharlal Road, Navrangpura,Ahmedabad Gujarat- 380009, 7874014621, hr@grownited.com

NEW L. J. INSTITUTE OF ENGINEERING AND TECHNOLOGY

DECLARATION

We hereby declare that the Internship report submitted along with the Internship entitled Project **The Urban Services** submitted in Complete for Bachelor of Engineering in **CSE(AIML)** Branch to **Gujarat Technological University**, Ahmedabad, is a bonafide record of original Internship work Completed by me at **Grownited Pvt Ltd** under the supervision of **External Guide Rahul Kirpekar** and that no part of this report has been directly copied from any students' reports or taken from any other source, without providing due reference.

Name of Student **Signature of Student**

Mr. Meet Rana

ACKNOWLEDGEMENT

I wish to express my sincere gratitude to my **External guide Rahul Kirpekar** for continuously guiding me at the company and answering all my doubts with patience. I would also like to thank **Dr. Gayatri Pandi (H.O.D. of CSE(AIML) / IT Department)** for motivating me every time whenever I get confused, I would also like to thank my **Internal Guide Prof. Mohit Patel** for helping me through my internship by giving me the necessary suggestions and advices along with their valuable co-ordination in completing this Internship.

I also thank my parents, friends and all the members of the family for their precious support and encouragement which they had provided in completion of my work. In addition to that, I would also like to mention the company personals who gave me the permission to use and experience the valuable resources required for the Internship.

Thus, in conclusion to the above said, I once again thank the staff members of **Grownited Pvt Ltd** for their valuable support in completion of the Internship.

Thank You

Name of Student **Date:**

Mr. Meet Rana

Enrollment Number **Signature of Student**

221433142029

TABLE OF CONTENTS

The Urban Services

Enrollment No: 221433142029

Student Name: Mr. Meet Rana

NEW L. J. INSTITUTE OF ENGINEERING AND TECHNOLOGY
(College Code: 143)

Semester: VIII, CSE(AIML) Department

ABSTRACT

The Urban Service platform is a comprehensive digital solution designed to connect urban residents with a wide range of essential services, such as plumbing, electrical repairs, cleaning, and carpentry. By creating a seamless interface between service providers and users, this platform simplifies the process of finding reliable professionals and ensures quality service delivery.

This project addresses the challenge of finding skilled and verified service providers in urban areas, offering convenience, transparency, and trust through a centralized system.

Keywords:

- **Urban Service Platform**
- **Digital Solution**
- **On-Demand Services**
- **Service Marketplace**
- **Home Maintenance Services**
- **Plumbing Services**
- **Electrical Repairs**

CHAPTER: 1

INTRODUCTION OF PROJECT & COMPANY PROFILE

1.1. Introduction

1.1.1. Company Profile

1.1.2. Company Products

1.1.3. Company Mission and Vision

1.2. Introduction of the Project

1.2.1. Purpose of the Project

1.2.2. Function Requirements

1.2.3. Problems in Existing System

1.2.4. Main Modules

1.1. Introduction

1.1.1. Company Profile

- Grownited Private Limited is a leading technology-driven company offering a full spectrum of IT services, including custom software development, web hosting, website development, mobile app development, e-commerce solutions, and branding services. Our expertise also extends to digital marketing, content creation, and UI/UX design, ensuring that businesses get comprehensive digital solutions under one roof.
- Founded with a vision to bridge the gap between technology and business success, Grownited Private Limited has grown into a trusted digital partner for numerous startups and enterprises. With a dedicated team of professionals, we have successfully delivered over 50 projects across various industries and continue to expand our presence globally. Our commitment to quality, innovation, and customer satisfaction sets us apart, making us the go-to choice for businesses looking to establish a strong digital presence.

1.1.2. Company Products

- **Grownited** is a leading technology solutions provider, specializing in **cutting-edge software development, IT training, and enterprise solutions**. Our goal is to empower individuals and businesses with **modern technologies and industry-specific expertise** to drive innovation and efficiency.
- **FARM Stack Development (FastAPI, React, MongoDB, JavaScript)**
  Grownited provides expert training and development services in **FARM Stack**, equipping learners and businesses with the skills to build **scalable, high-performance web applications**. Our hands-on approach ensures proficiency in backend development with **FastAPI**, frontend design with **React.js**, and **MongoDB** for efficient data management.

- **Joomla CMS Customization & Development :**
  Joomla is a **powerful, flexible, and secure** Content Management System (CMS). At Grownited, we offer **Joomla customization and training**, enabling businesses to enhance their websites, integrate new modules, and add plugins to meet unique needs. Our structured approach ensures seamless implementation and optimized web performance.
- **Enterprise Software Development & Project Training**
  We provide **end-to-end enterprise software solutions**, helping businesses implement **scalable tools for planning, operations, and digital transformation**. Our **project training** focuses on equipping professionals with **real-world development experience** and best industry practices.
- **Comprehensive IT Training & Placement Services**
  Grownited offers **in-depth training programs** in various **programming and non-programming skills**, including:

Characteristics

- **FastAPI And MongoDb Development**
- **Java & ASP.Net**
- **Mobile App Development**
- **Digital Marketing & SEO**
- **Graphic & Web Designing**

- Our **industry-focused training** is led by expert instructors, featuring **live project-based learning** to ensure real-world exposure. We also offer **placement services** to help freshers and experienced professionals secure job opportunities in Ahmedabad's growing IT sector.
- At **Grownited**, we are committed to delivering **innovative IT solutions and training programs** that empower businesses and individuals to **thrive in the digital era**.

1.1.3. Company Mission and Vision

Company Mission:

- To provide **high-quality, practical, and industry-relevant** IT training that empowers individuals with **job-ready skills**.
- Grownited aims to deliver **innovative and scalable web solutions**, enabling businesses to **harness the full potential** of modern technologies.
- We ensure that our **development and training services** contribute to enhancing the **efficiency, performance, and growth** of businesses.
- **Our training programs are structured to help candidates** seamlessly transition into the industry **with hands-on experience in** real-world projects**.**

Company Vision:

- To become a **leading IT solutions provider and training center**, recognized for **excellence in technology innovation and education**.
- To create a **platform for aspiring tech professionals** to learn, grow, and thrive in the fast-evolving IT industry.
- To foster **customer-centric development**, ensuring businesses receive **cutting-edge and customized solutions** for sustainable growth.
- To bridge the **gap between learning and employment** by offering **industry-standard training and placement support**.

1.2. Introduction of the Project

1.2.1. Purpose of the Project

- Ensure proper scheduling and uninterrupted workflow for urban service operations, minimizing delays and completing tasks on time.
- Efficiently assign tasks, allocate resources, and define responsibilities for optimized service delivery.
- Establish a clear understanding of roles, timelines, and costs associated with various urban services.
- Integrate all service-related tasks to enhance quality, improve efficiency, and reduce costs for clients.
- Accelerate service processing and response times.
- Enable task completion with selected personnel or service providers as per requirements.
- Provide a streamlined selection process for assigning available personnel, ensuring workforce availability when needed.
- Maintain security and confidentiality of sensitive urban service data and operational information.

1.2.2. Function Requirements

- **Task Management:** Create, update, delete, assign tasks, set priorities, and deadlines.
- **Task Tracking:** Monitor task status, history, and generate reports.
- **Collaboration:** Send task notifications, emails, and allow reporting to managers/team leaders.
- **Time Management:** Maintain timesheets for work processes and track employee work hours.
- **Reporting & Analytics:** Generate daily reports, task summaries, and performance insights.

1.2.3. Problems in Existing System

- **Time-Consuming Searches:** Finding necessary information requires **manual searching**, leading to delays.
- **Inefficient Communication:** Information transfer relies on **physical documents or letters**, slowing down workflow.
- **Delays in Drafting Letters:** Preparing and sending letters takes significant time, reducing productivity.

1.2.4. Main Modules

- **Admin Side**
    - Manage The Employees.
    - Edit the functionality
        - Add Task
        - Delete Task
        - Update Task
        - Changing the information like priority, Duration, Start date, End date
    - Assign the Task to Resources.
    - Advance Search
        - By Task ID
        - By Employee Name
        - By Employee ID
    - Show the Recent Activity.
    - Show the List of Projects.
- **User Side**
    - Employee can Login Properly with Employee Id & Password.
    - Update the Status
        - In progress
        - Completed
        - Close
        - If any query, then Write comments
    - Show the Recent Activity.
    - Show the List of Tasks.
    - Attach the Files.
    - Submit Task.

CHAPTER: 2

SYSTEM REQUIREMENTS

2.1 Hardware & Software Requirements

2.1.1 Server-Side Requirements

2.1.2 Developer Side Requirements

2.1.3 User Side Requirements

2.1 Hardware & Software Requirements

2.1.1 Server-Side Requirements

- **Hardware Requirement**
  - I3 Processor or above.
  - 500 GB HDD or above
  - 8GB of RAM.
  - Keyboard (Normal or Multimedia) and Mouse (Compatible Mouse)
- **Software Requirement**
  - **Front End**: React, Material UI and JavaScript
  - **Back End:** FastAPI, Mongo DB and React Backend
  - **Server:** NPM and Uvicorn
  - **Design Tool:** Sublime, VS Code
  - **OS:** Window 7 or Any Compatible OS

2.1.2 Developer Side Requirements

- **Hardware Requirement**
  - I3 Processor or above.
  - 500 GB HDD or above
  - 8GB of RAM.
  - Keyboard (Normal or Multimedia) and Mouse (Compatible Mouse)
- **Software Requirement**
  - **Front End**: React, Material UI and JavaScript
  - **Back End:** FastAPI, Mongo DB, React Backend
  - **Server:** NPM and Uvicorn
  - **Design Tool:** Sublime, VS Code
  - **OS:** Window 7 or Any Compatible OS

2.1.3 User Side Requirements

- **Hardware Requirement**
  - I3 Processor or above.
  - 500 GB HDD or above
  - 8GB of RAM.

- **Software Requirement**
  - O**S:** Window 7 or Any Compatible OS
  - **Browser:** Chrome or any Compatible browser
- **Basic Requirement**
  - Internet Connection.
- **Front End**: React, Material UI, and JavaScript
- **Back End:** Fast API, Mongo DB, React-Backend, NPM and Uvicorn

CHAPTER: 3

WORK SHEET REPORT

3.1 Work Sheet Report (15 DAYS)

SUGGESTED 15 DAYS WORK SHEET REPORT				
Student Name:	**Mr. Meet Rana**			
Enrollment No:	**221433142029**			
Internship/Project Title	**The Urban Services**			
Tools and Technologies	**Front End:** React, Material UI and JavaScript **Back End:** FastAPI, Mongo DB, React Backend **Server:** NPM and Uvicorn **Design Tool:** VS Code			
Company/ Organization Name	**Grownited Pvt Ltd**			
Student's Activity Details:				
Week Number	**Start Date to End Date**	**Tasks to be assigned**	**Tasks to be completed**	**Remarks**
1	15/01/2025 To 20/01/2025	Requirement Analysis, Data Dictionary System Design	Completed	
2	22/01/2025 To 27/01/2025	Front-end: React Tags	Completed	

SUGGESTED 15 DAYS WORK SHEET REPORT

Student Name:	**Mr. Meet Rana**
Enrollment No:	**221433142029**
Internship/Project Title	**The Urban Services**
Tools and Technologies	**Front End:** React, Material UI and JavaScript **Back End:** FastAPI, Mongo DB, React Backend **Server:** NPM and Uvicorn **Design Tool:** VS Code
Company/ Organization Name	**Grownited Pvt Ltd**

Student's Activity Details:

Week Number	Start Date to End Date	Tasks to be assigned	Tasks to be completed	Remarks
3	29/01/2025 To 03/02/2025	Front-end: MUI	Completed	
4	05/02/2025 To 10/02/2025	Front-end: Core JavaScript	Completed	

SUGGESTED 15 DAYS WORK SHEET REPORT

Student Name:	**Mr. Meet Rana**
Enrollment No:	**221433142029**
Internship/Project Title	**The Urban Services**
Tools and Technologies	**Front End:** React, Material UI and JavaScript **Back End:** FastAPI, Mongo DB, React Backend **Server:** NPM and Uvicorn **Design Tool:** VS Code
Company/ Organization Name	**Grownited Pvt Ltd**

Student's Activity Details:

Week Number	Start Date to End Date	Tasks to be assigned	Tasks to be completed	Remarks
5	12/02/2025 To 17/02/2025	Front-end: Advance JavaScript	Completed	
6	19/02/2025 To 24/02/2025	Front-end: Java Script	Completed	

SUGGESTED 15 DAYS WORK SHEET REPORT

Student Name:	**Mr. Meet Rana**
Enrollment No:	**221433142029**
Internship/Project Title	**The Urban Services**
Tools and Technologies	**Front End:** React, Material UI and JavaScript **Back End:** FastAPI, Mongo DB, React Backend **Server:** NPM and Uvicorn **Design Tool:** VS Code
Company/ Organization Name	**Grownited Pvt Ltd**

Student's Activity Details:

Week Number	Start Date to End Date	Tasks to be assigned	Tasks to be completed	Remarks
7	26/02/2025 To 02/03/2025	Template Create Using Frontend	Completed	
8	04/03/2025 To 09/03/2025	Back-end: Core FastAPI	Completed	

SUGGESTED 15 DAYS WORK SHEET REPORT

Student Name:	**Mr. Meet Rana**
Enrollment No:	**221433142029**
Internship/Project Title	**The Urban Services**
Tools and Technologies	**Front End:** React, Material UI and JavaScript **Back End:** FastAPI, Mongo DB, React Backend **Server:** NPM and Uvicorn **Design Tool:** VS Code
Company/ Organization Name	**Grownited Pvt Ltd.**

Student's Activity Details:

Week Number	Start Date to End Date	Tasks to be assigned	Tasks to be completed	Remarks
9	11/03/2025 To 16/03/2025	Back-end: Fast API With Python	Completed	
10	18/03/2025 To 23/03/2025	Back-end: Mongo DB & Table Schema	Completed	

SUGGESTED 15 DAYS WORK SHEET REPORT

Student Name:	**Mr. Meet Rana**
Enrollment No:	**221433142029**
Internship/Project Title	**The Urban Services**
Tools and Technologies	**Front End:** React, Material UI and JavaScript **Back End:** FastAPI, Mongo DB, React Backend **Server:** NPM and Uvicorn **Design Tool:** VS Code
Company/ Organization Name	**Grownited Pvt Ltd.**

Student's Activity Details:

Week Number	**Start Date to End Date**	**Tasks to be assigned**	**Tasks to be completed**	**Remarks**
11	25/03/2025 To 30/03/2025	Project Implementation	Completed	
12	01/04/2025 To 06/04/2025	Project Implementation	Completed	

SUGGESTED 15 DAYS WORK SHEET REPORT

Student Name:	**Mr. Meet Rana**
Enrollment No:	**221433142029**
Internship/Project Title	**The Urban Services**
Tools and Technologies	**Front End:** React, Material UI and JavaScript **Back End:** FastAPI, Mongo DB, React Backend **Server:** NPM and Uvicorn **Design Tool:** VS Code
Company/ Organization Name	**Grownited Pvt Ltd.**

Student's Activity Details:

Week Number	Start Date to End Date	Tasks to be assigned	Tasks to be completed	Remarks
13	08/04/2025 To 13/04/2025	Project Implementation	Completed	
14	15/04/2025 To 20/04/2025	Project Implementation	Completed	

CHAPTER: 4

FRONT END OF SYSTEM

4.1 About Front End

4.1.1 about React

4.1.2 about Material UI

4.1.3 about JavaScript

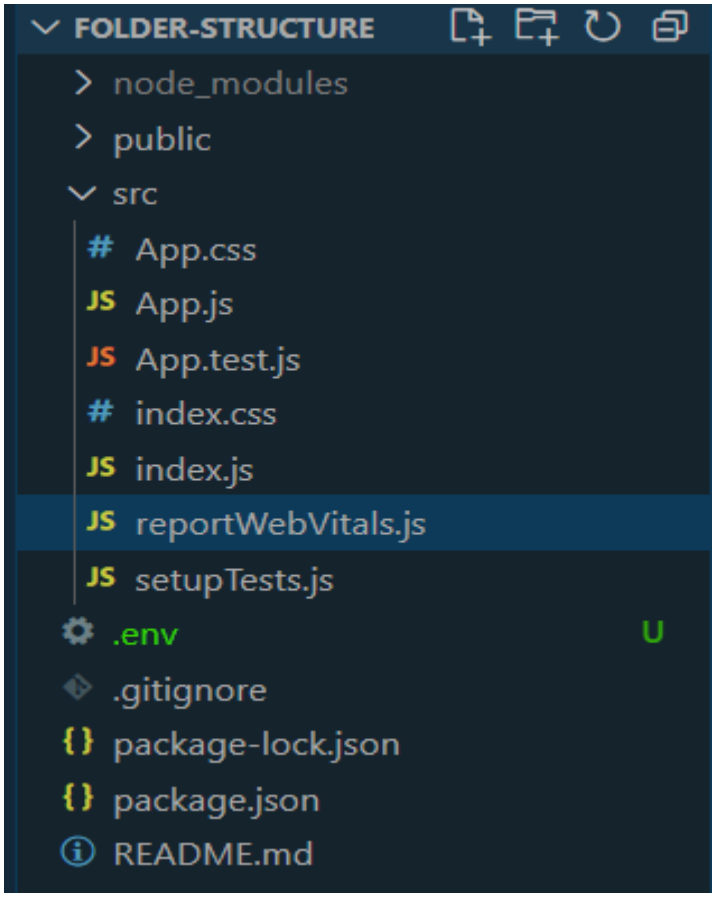
4.1 About Front End

- Front end development refers to the part of web development that focuses on creating the user interface and experience of a website or web application.
- It involves implementing designs provided by UI/UX designers using languages such as React, Material UI and JavaScript.

4.1.1 About React

- React is a JavaScript library for building UI components.
- It uses a component-based architecture for reusability..
- JSX allows writing HTML-like syntax within JavaScript.
- Virtual DOM improves rendering efficiency.
- Components can be functional or class-based.
- React manages state and UI updates efficiently.

React File Structure:



- The logic includes state management, hooks, and event handling.
- It defines how data is processed and interacts with the component.
- The UI is defined using JSX, which structures the visual representation.
- React, a JavaScript library, is used to build interactive web applications.

- **Here's a breakdown of React key components and concepts:**

Components:

- React applications are built using components, which are reusable UI blocks
- Components can be functional or class-based and manage their own logic.

JSX (JavaScript XML):

- JSX allows writing HTML-like syntax inside JavaScript.
- It makes component structures more readable and expressive.

Props (Properties):

- Props are used to pass data from one component to another.
- They are immutable and help make components reusable.

State Management:

- State holds dynamic data and controls component behavior.
- It can be managed using the `useState` hook in functional components.

Component Lifecycle:

- React components go through different lifecycle phases: mounting, updating, and un-mounting**.**

- The `useEffect` hook allows handling side effects like API calls and subscriptions.

Event Handling:

- React uses event listeners like `onClick`, `onChange`, and `onSubmit` for interactivity.
- Events in React follow a synthetic event system for cross-browser consistency.

Routing and Navigation:

- React Router enables navigation between different pages without reloading the browser.
- It uses `Route`, `Link`, and `useNavigate` for seamless navigation.
- React provides a powerful way to build interactive web applications.
- It is often combined with CSS for styling and JavaScript for logic to create dynamic UIs.

4.1.2 About Material-UI:

- **MUI is a Popular React UI Frame Work** Based on the Google Material Design Principles.
- **MUI provides pre-styled React components** that enhance UI consistency and reduce development time
- **MUI supports themes and customization,** allowing developers to style components efficiently
- **MUI components follow accessibility standards** to ensure a better user experience
- **MUI integrates with CSS-in-JS libraries** like Emotion for styling flexibility.
- **Key Concepts of Material-UI:**

Components:

- MUI provides a collection of pre-built UI components like `Button`, `Card`, `Typography`, and `Dialog`.
- These components follow Material Design guidelines for a modern look and feel.

Styling and Theming:

- MUI uses the `sx` prop, styled components, and custom themes for styling.
- Developers can override default styles using the `ThemeProvider`.

- **Grid and Layout System:**
  - MUI includes a powerful `Grid` and `Box` system for responsive layouts rules of specificity, which determine which styles are applied when multiple rules conflict.

- **Pre-built Components and Icons:**
  - Includes a rich set of **UI components** like buttons, modals, tables, and cards, reducing development time.
  - **Material Icons** are integrated for a seamless design experience.
  - MUI simplifies front-end development by offering **customizable, responsive, and accessible** UI components that enhance user experience in React applications..

4.1.3 about JavaScript

- JavaScript is a high-level, versatile programming language primarily known for its use in web development.
- It enables developers to add interactivity, behavior, and dynamic features to websites and web applications. Here's a comprehensive overview of JavaScript:

Client-Side Scripting:

- JavaScript is primarily used as a client-side scripting language, meaning it runs on the user's web browser rather than a remote server.
- This allows for dynamic updates and interaction without needing to reload the entire webpage.

Core Features:

Variables and Data Types:

- JavaScript supports various data types such as numbers, strings, booleans, arrays, objects, functions, and more.
- Variables are used to store data, and they can be dynamically typed.

Control Structures:

- JavaScript provides control structures like conditionals (if, else if, else), loops (for, while, do-while), and switch statements.

Functions:

- Functions are reusable blocks of code that perform a specific task.
- JavaScript supports both function declarations and function expressions, as well as arrow functions introduced in ES6.

Objects and Prototypes:

- JavaScript is an object-oriented language, where objects are collections of key-value pairs.
- Objects can also inherit properties and methods from other objects through prototypes.

DOM Manipulation:

- The Document Object Model (DOM) represents the structure of HTML documents as a tree of objects.
- JavaScript can manipulate the DOM to dynamically change the content, structure, and style of web pages.

Event-Driven Programming:

- JavaScript follows an event-driven programming paradigm, where actions or events (like mouse clicks, key presses, or page loads) trigger the execution of specific code.

- Event handlers are used to respond to these events and perform actions accordingly.

Asynchronous Programming:

- JavaScript supports asynchronous programming through mechanisms like callbacks, promises, and async/await.
- This allows non-blocking execution of code, enabling tasks like fetching data from servers, handling user input, and performing animations without freezing the UI.

Libraries and Frameworks:

- JavaScript has a vast ecosystem of libraries and frameworks that extend its capabilities and streamline development. Popular libraries include jQuery for DOM manipulation and AJAX requests, while frameworks like React.js, Angular, and Vue.js provide structured approaches to building complex web applications.

Server-Side Development:

- While JavaScript is primarily associated with client-side scripting, it is also used for server-side development.
- Node.js is a runtime environment that allows JavaScript to be executed on the server, enabling full-stack JavaScript development.

Security Considerations:

- JavaScript runs in a sandboxed environment within the browser, which limits its access to the user's system.
- However, developers must still be cautious of security vulnerabilities like cross-site scripting (XSS) and ensure proper data validation and sanitization.
- JavaScript is a fundamental technology for web development, empowering developers to create interactive and engaging web experiences across various devices and platforms.
- Its versatility and extensive ecosystem make it a cornerstone of modern web development.

CHAPTER: 5

BACK END OF SYSTEM

5.1 About Fast-API

5.2 Why Use Fast-API

5.3 About Back End

5.3.1 about Mongo-DB

5.3.2 How Mongo-DB Works

5.3.3 Mongo-DB Features

5.4 Why use Mongo-DB

5.1 About Fast-API

FastAPI is a **high-performance**, **asynchronous** web framework for building **fast** and **scalable** APIs in Python. It is built on **Starlette** for web handling and **Pydantic** for data validation, making it an ideal choice for modern API development.

High-Performance API Framework:

- **FastAPI** is a modern, high-performance web framework for building APIs with Python 3.7+ based on **Starlette** and **Pydantic**.
- It is designed to handle **asynchronous requests efficiently**, making it one of the fastest Python frameworks.

Asynchronous and Synchronous Support:

- Supports both **Async and Sync** request handling, leveraging Python's **Async IO** for non-blocking I/O operations.
- Ideal for building **real-time applications** and handling high concurrency.

Automatic Data Validation and Serialization:

- Uses **Pydantic** for automatic **request validation, data parsing, and serialization**.
- Type annotations ensure **fewer errors and better code maintainability**.

Interactive API Documentation:

- **Built-in Swagger UI and REDOC** allow developers to test API endpoints effortlessly.
- **Auto-generated OPENAPI schema** simplifies API documentation.

Dependency Injection & Security Features:

- Provides an intuitive **dependency injection system** for better modularity and reusability.
- Includes built-in security features like **OAuth2, JWT authentication, and API key handling**.

Database and ORM Support:

- Compatible with multiple **databases**, including **PostgreSQL, MongoDB, Mongo DB, and SQLite**.
- Works well with **SQLAlchemy, Tortoise-ORM, and Beanie** for seamless database interactions.

Scalable and Production-Ready:

- Designed for **microservices** and **large-scale applications** with minimal boilerplate code.
- Optimized to work efficiently with **Docker and Kubernetes** for deployment

5.2 Why Use FastAPI

FastAPI is an **advanced, high-performance web framework** designed to build APIs quickly and efficiently using Python. Here's why developers and companies prefer FastAPI over other frameworks.

- **Asynchronous Support**: Uses Python's **async/await** for handling thousands of requests simultaneously.
- **Speed Comparable to Node.js & Go**: Thanks to Starlette and Pydantic, it's one of the fastest Python frameworks.
- **Pydantic-Based Validation**: Ensures data correctness before processing requests.
- **Type Hints for Data Handling**: Uses Python type hints (`str`, `int`, `List`, `Dict`) to automatically validate input.
- **Swagger UI & ReDoc**: API documentation is automatically generated, making testing and integrationeasier.
- **Interactive API Testing**: Developers can test endpoints directly from the browser.
- **Faster Development**: Write less code while achieving more functionality. **Clean and Readable Code**: Uses Pythonic syntax, making development intuitive.
- **OAuth2, JWT, API Keys, and Basic Auth** support for authentication. **Prevent common vulnerabilities** like SQL Injection and CSRF with built-in protections.
- Works with **SQL (PostgreSQL, Mongo DB, SQLite) and NoSQL (MongoDB, Firebase, etc.)**
- Supports popular ORMs like **SQLAlchemy, Tortoise-ORM, and Beanie (MongoDB)**.
- **Docker & Kubernetes Ready**: Works seamlessly in cloud and microservices environments. **Asynchronous Requests**: Allows handling large-scale applications efficiently.

5.3 About Back End

The backend, also known as server-side, refers to the part of a web application or software that operates behind the scenes and is responsible for processing requests, managing data, and generating responses to be sent to the client-side (frontend). Here are some key aspects of backend development:

- **Data Management:** One of the primary functions of the backend is managing data. This includes tasks such as storing, retrieving, updating, and deleting data from databases. Backend developers work with databases such as Mongo DB, PostgreSQL, MongoDB, or others to ensure efficient data storage and retrieval.
- **Business Logic:** The backend contains the business logic of the application, which defines how data is processed and manipulated based on the application's rules and requirements. This may involve implementing algorithms, calculations, validations, and other operations necessary to achieve the desired functionality.
- **Server-Side Scripting:** Backend development often involves writing server-side code using programming languages like PHP, Python, Ruby, Java, Node.js, or others. This code runs on the server and handles client requests, executes business logic, interacts with databases, and generates dynamic content.
- **APIs (Application Programming Interfaces):** Backend developers design and implement APIs that allow different components of a system to communicate with each other. APIs define the protocols and rules for interacting with the backend, enabling frontend applications, mobile apps, and other services to access and manipulate data.

- **Security:** Backend developers are responsible for implementing security measures to protect the application and its data from unauthorized access, data breaches, and other security threats. This includes techniques such as authentication, authorization, encryption, input validation, and secure coding practices.
- **Scalability and Performance:** Backend developers design and optimize the backend architecture to ensure scalability and performance, especially for applications with high traffic or large datasets. This may involve using caching mechanisms, load balancing, horizontal scaling, and other techniques to handle increased demand and maintain responsiveness.
- **Integration:** Backend systems often need to integrate with external services, third-party APIs, payment gateways, and other systems. Backend developers handle the integration process, ensuring seamless communication and interoperability between different components of the application ecosystem.
- **Monitoring and Maintenance:** Backend developers monitor the health and performance of the backend systems, identify and troubleshoot issues, and perform routine maintenance tasks such as database backups, software updates, and security patches.

In summary, backend development is crucial for building robust, scalable, and secure web applications and software systems. It involves handling data management, implementing business logic, writing server-side code, designing APIs, ensuring security, optimizing performance, integrating with external services, and maintaining the health of backend systems.

5.3.1 about Mongo DB

Mongo DB is an open-source relational database management system (RDBMS) that is widely used for web development and other applications requiring a robust, scalable, and reliable database solution. It's known for its speed, ease of use, and comprehensive support for various programming languages and platforms.

Here are some key aspects of Mongo DB:

- **Relational Database Management System:** Mongo DB follows the relational database model, organizing data into tables with rows and columns. It supports SQL (Structured Query Language) for querying and managing the database.
- **Open Source:** Mongo DB is an open-source software, meaning it is freely available for anyone to use, modify, and distribute. This has contributed to its widespread adoption and vibrant community of developers and contributors.
- **Cross-Platform Compatibility:** Mongo DB is compatible with multiple operating systems, including Windows, macOS, Linux, and various Unix variants. This flexibility allows developers to deploy Mongo DB on a wide range of platforms.
- **Scalability and Performance:** Mongo DB is designed to be highly scalable, capable of handling large volumes of data and high traffic loads. It offers features such as replication, sharding, and clustering to distribute data and workload across multiple servers for improved performance and reliability.
- **Data Security:** Mongo DB provides robust security features to protect data integrity and confidentiality. This includes user authentication, access control, encryption, and auditing capabilities to safeguard sensitive information from unauthorized access and attacks.
- **High Availability:** Mongo DB supports various high availability solutions, such as master-slave replication, multi-master replication, and clustering, to ensure continuous availability and fault tolerance. These features help minimize downtime and maintain data consistency in case of hardware failures or network issues.
- **Community and Support:** Mongo DB has a large and active community of developers, users, and contributors who provide support, share knowledge, and contribute to the improvement of the software. Additionally, commercial support and consulting services are available from Mongo DB vendors for organizations that require professional assistance.

- **Integration and Compatibility:** Mongo DB integrates seamlessly with popular web development technologies and frameworks, including PHP, Python, Ruby on Rails, Node.js, and Java. It also supports standard database interfaces such as ODBC and JDBC, enabling interoperability with a wide range of applications and tools.
- Overall, Mongo DB is a powerful and versatile database management system that is well-suited for a variety of applications, from small-scale websites to large enterprise systems. Its combination of performance, scalability, reliability, and ease of use makes it a popular choice among developers and organizations worldwide.

5.3.2 How Mongo DB Works

Mongo DB works as a relational database management system (RDBMS) that stores and manages structured data organized into tables. Here's an overview of how Mongo DB works:

- **Data Organization:** Mongo DB organizes data into tables, where each table consists of rows and columns. Each column represents a specific attribute of the data, while each row contains a single record or tuple with values for those attributes.
- **SQL Interface:** Mongo DB uses SQL (Structured Query Language) as its interface for interacting with the database. SQL allows users to perform various operations such as querying data, inserting new records, updating existing records, and deleting records.
- **Client-Server Architecture:** Mongo DB follows a client-server architecture, where multiple clients can connect to a Mongo DB server to access and manipulate data. The Mongo DB server manages database operations and handles requests from clients, executing SQL queries and returning results as requested.
- **Connection Handling:** Clients connect to the Mongo DB server over a network using TCP/IP or Unix sockets. Each client connection is authenticated using a username and password, and the server enforces access control based on user privileges and permissions.

- **Query Processing:** When a client sends a SQL query to the Mongo DB server, the server parses the query, creates an execution plan, and executes the query against the database tables. This involves retrieving data from disk storage, performing any necessary computations or joins, and returning the results to the client.
- **Storage Engines:** Mongo DB supports multiple storage engines, each optimized for different types of workloads and data access patterns. The default storage engine for Mongo DB is InnoDB, which provides features such as ACID (Atomicity, Consistency, Isolation, Durability) transactions, foreign key constraints, and crash recovery.
- **Indexing:** Mongo DB uses indexes to optimize query performance by speeding up data retrieval operations. Indexes are data structures that store a sorted copy of selected columns from a table, allowing the server to quickly locate and access the rows that match a specific search condition.
- **Concurrency Control:** Mongo DB employs concurrency control mechanisms to ensure data consistency and integrity in multi-user environments. This includes techniques such as locking, multi-version concurrency control (MVCC), and isolation levels to manage concurrent access to shared data and prevent conflicts between transactions.
- **Transaction Management:** Mongo DB supports transactions, which are sequences of SQL statements that are executed as a single unit of work. Transactions ensure that database changes are atomic, consistent, isolated, and durable (ACID properties), even in the event of system failures or interruptions.

- Overall, Mongo DB works by providing a reliable, efficient, and scalable platform for storing, managing, and accessing structured data, making it suitable for a wide range of applications and use cases.

5.3.3 Mongo DB Features

Mongo DB offers a rich set of features that make it a popular choice for building database-driven applications. Here are some key features of Mongo DB:

- **Cross-Platform Compatibility:** Mongo DB is compatible with various operating systems, including Windows, macOS, Linux, and Unix variants. This allows developers to deploy Mongo DB on a wide range of platforms and environments.
- **Scalability:** Mongo DB is designed to scale from small, single-server deployments to large, distributed systems handling millions of transactions per second. It supports features such as replication, sharding, and clustering to distribute data and workload across multiple servers for improved performance and scalability.
- **High Availability:** Mongo DB offers several features for ensuring high availability and fault tolerance, including master-slave replication, multi-master replication, and clustering. These features help minimize downtime and ensure continuous availability of data and services.
- **Security:** Mongo DB provides robust security features to protect data integrity and confidentiality. This includes user authentication, access control, encryption, and auditing capabilities to safeguard sensitive information from unauthorized access and attacks.
- **Performance Optimization:** Mongo DB includes various performance optimization features to improve query performance and throughput. This includes indexing, query caching, query optimization, and support for advanced storage engines such as InnoDB and MyISAM.
- **Data Types:** Mongo DB supports a wide range of data types, including numeric, string, date and time, spatial, and JSON data types. This allows developers to store and manipulate different types of data efficiently.
- **Stored Procedures and Functions:** Mongo DB supports stored procedures and user-defined functions, allowing developers to encapsulate SQL logic into reusable code blocks. This improves code modularity, maintainability, and performance.

- **Triggers and Events:** Mongo DB supports triggers and events, which are database objects that automatically execute in response to specified events or conditions. Triggers can be used to enforce data integrity constraints, audit changes, or perform other actions.
- **Transaction Support:** Mongo DB provides support for transactions, which are sequences of SQL statements that are executed as a single unit of work. Transactions ensure that database changes are atomic, consistent, isolated, and durable (ACID properties), even in the event of system failures or interruptions.
- **Backup and Recovery:** Mongo DB includes features for backup and recovery, allowing users to create full or incremental backups of databases and restore them in the event of data loss or corruption.

Overall, Mongo DB offers a comprehensive set of features for building scalable, high- performance, and secure database-driven applications, making it a popular choice for developers and organizations worldwide.

5.4 Why use Mongo DB

There are several reasons why Mongo DB is a popular choice for building database-driven applications:

- **Performance:** Mongo DB is known for its high performance and efficiency, making it well- suited for applications that require fast and responsive data access. It supports various optimization techniques such as indexing, caching, and query optimization to improve query performance and throughput.
- **Scalability:** Mongo DB is highly scalable and can handle large volumes of data and high traffic loads. It supports features like replication, sharding, and clustering to distribute data and workload across multiple servers, allowing applications to scale horizontally as demand grows.
- **Reliability:** Mongo DB is a stable and reliable database management system with a proven track record in production environments. It provides features for ensuring data integrity, transaction consistency, and fault tolerance, making it suitable for mission-critical applications.

- **Cross-Platform Compatibility:** Mongo DB is compatible with various operating systems and platforms, including Windows, Mac OS, Linux, and Unix variants. This allows developers to deploy Mongo DB on different environments without significant modifications to their applications.
- **Open Source:** Mongo DB is open-source software, meaning it is freely available for anyone to use, modify, and distribute. This makes it a cost-effective choice for businesses and developers, as there are no licensing fees associated with using Mongo DB.
- **Community and Support:** Mongo DB has a large and active community of developers, users, and contributors who provide support, share knowledge, and contribute to the improvement of the software. Additionally, commercial support and consulting services are available from Mongo DB vendors for organizations that require professional assistance.
- **Integration:** Mongo DB integrates seamlessly with popular programming languages, frameworks, and tools used in web development, such as PHP, Python, Ruby on Rails, Node.js, and Java. It also supports standard database interfaces such as ODBC and JDBC, enabling interoperability with a wide range of applications and tools.
- **Security**: Mongo DB provides robust security features to protect data integrity and confidentiality. This includes user authentication, access control, encryption, and auditing capabilities to safeguard sensitive information from unauthorized access and attacks.

Overall, Mongo DB offers a combination of performance, scalability, reliability, ease of use, and community support that makes it a preferred choice for building database-driven applications of all sizes and complexity levels.

CHAPTER 6
SYSTEM ANALYSIS AND SYSTEM DESIGN

6.1 System Analysis and Design

6.2 Systems Development Life Cycle (SDLC)

6.1 System Analysis and Design

System Analysis and Design (SAD) is the backbone of software development, guiding the creation of efficient and effective systems to address organizational needs. At its core, SAD is about understanding the current state of affairs within an organization, envisioning a better future, and architecting the pathway to get there.

During the analysis phase, SAD professionals delve deep into the existing systems and processes, meticulously identifying strengths, weaknesses, and opportunities for improvement. Through interviews, surveys, and observation, they gather requirements from stakeholders, ensuring that every voice is heard and every need is accounted for.

With requirements in hand, the design phase begins, where SAD experts transform abstract ideas into concrete plans. They define the system architecture, user interfaces, and data structures, ensuring that the design aligns closely with the gathered requirements and the overarching goals of the organization.

Implementation brings the design to life, as developers translate the blueprints into functional code. Rigorous testing follows, with each component scrutinized to ensure its reliability, security, and performance. User acceptance testing ensures that the system not only meets technical specifications but also satisfies the needs and expectations of its intended users.

Deployment marks the culmination of the SAD process, as the new system is introduced into the organization's ecosystem. With proper training and support, users adapt to the new system, embracing its capabilities and maximizing its potential to drive efficiency and innovation.

But the journey doesn't end there. Maintenance and support are ongoing responsibilities, as SAD professionals monitor the system, address issues, and incorporate updates to keep it running smoothly and meeting the evolving needs of the organization.

Throughout every phase of System Analysis and Design, collaboration, communication, and a relentless focus on the end-users are paramount. By aligning technology with business objectives and user needs, SAD professionals play a vital role in shaping the future success of organizations across industries.

6.2 Systems Development Life Cycle (SDLC)

A software life cycle model (also termed process model) is a pictorial and diagrammatic representation of the software life cycle. A life cycle model represents all the methods required to make a software product transit through its life cycle stages. It also captures the structure in which these methods are to be undertaken.

In other words, a life cycle model maps the various activities performed on a software product from its inception to retirement. Different life cycle models may plan the necessary development activities to phases in different ways. Thus, no element which life cycle model is followed, the essential activities are contained in all life cycle models though the action may be carried out in distinct orders in different life cycle models. During any life cycle stage, more than one activity may also be carried out.

CHAPTER: 7

DATA DICTIONARY

7.1 Introduction

7.2 List of Tables

7.2.1 : Admin_master

7.2.2 : User_ master

7.2.3 : Task _master

7.2.4 : Employee_master

7.2.5 : Task_Assignment_master

7.2.6 : Recent_Activity_master

7.2.7 : Project _master

7.1 Introduction

Data Dictionary is an important part of a project or system which contains all definition of elements in the system. In Data Dictionary you will find a list of all elements composing the data flowing through a system.

- The major elements of a system:
  - Data Flow
  - Data Store
  - Processes
- The data dictionary stores all details and description of these elements.
- The data dictionary provides additional information about the system.
- The data dictionary contains the data about data e.g. Student is a data and this student belongs to this course is a description of STUDENT data which is stored in data dictionary.
- Why is Data Dictionary important?
  - To manage the details in large system.
  - To communicate a common meaning for all system elements.
  - To document the features of system.
  - To determine where system changes should made.
  - To locate errors and omissions in the system.

7.2 List of Tables:

- **Table 7.2.1: End User Profile**

  Description: This table stores data related to users of the ride-sharing platform, including their personal details, ride preferences, and travel history to facilitate seamless ride coordination and user management**.**

	End User Profile	
Field Name	Type	Description
userId	**INT (PK)**	**Unique identifier for each user**
firstName	**VARCHAR(100)**	**User's first name**
lastName	**VARCHAR(100)**	**User's last name**
email	**VARCHAR(100)**	**Unique email address**
password	**VARCHAR(255)**	**Encrypted password**
gender	**ENUM('Male', 'Female', 'Other')**	**User's gender**
contactNum	**VARCHAR(15)**	**Contact number**
role	**ENUM(1,2,3)**	**1 = Admin, 2 = Customer, 3 = Service Provider**
createdAt	**TIMESTAMP**	**Account creation timestamp**
status	**ENUM(1,2)**	**1 = Active, 2 = Disabled**

[Table 7.2.1: End User Profile]

Table 7.2.2: SERVICE PROVIDERS PROFILE

Description: This table stores data related to drivers or ride service providers, including their personal details, vehicle information, availability, and ride history to ensure efficient ride allocation and management.

	SERVICE PROVIDERS PROFILE	
Filed Name	**Type**	**Description**
providerId	**Unique service provider identifier**	**Unique service provider identifier**
providerName	**Name of the provider**	**Name of the provider**
UserId	**Foreign key linking to User**	**Foreign key linking to User**
ServiceId	**Foreign key linking to Service**	**Foreign key linking to Service**
subServiceId	**Foreign key linking to subService**	**Foreign key linking to subService**
businessAddress	**Business address**	**Business address**
latitude	**Geographic latitude**	**Geographic latitude**
longitude	**Geographic longitude**	**Geographic longitude**
availability	**Available service timings**	**Available service timings**
rating	**Average service rating**	**Average service rating**
status	**1 = Active, 2 = Inactive**	**1 = Active, 2 = Inactive**

[Table 7.2.2: Service Providers Profile]

Table 7.2.3: ADMIN PROFILE

Description: This table stores data related to platform administrators, including their roles, permissions, and system management activities to oversee and maintain the smooth operation of the ride-sharing platform.

	ADMIN PROFILE	
Field Name	**Type**	**Description**
AdminId	**INT (PK)**	**Unique identifier**
AdminName	**VARCHAR(100)**	**Name of the admin**
ProviderId	**INT (FK)**	**Foreign key linking to Provider**
UserId	**INT (FK)**	**Foreign key linking to User**
ServiceId	**INT (FK)**	**Foreign key linking to Service**
subServiceId	**INT (FK)**	**Foreign key linking to subService**
ProviderId + Service_Id	**INT(FK) + INT(FK)**	**Foregin Key Of Provider Id With Assign Service_id**
UserId + Service_Id	**INT(FK) _ INT(FK)**	**Foregin Key Of UserId With Assign Service_id**

[Table 7.2.3: Admin _Profile**]**

Table 7.2.4: SERVICES BOOKINGS

Description: This table stores information about employees.

	SERVICES BOOKINGS	
Field Name	**Type**	**Description**
bookingId	**INT (PK)**	**Unique booking identifier**
userId	**INT (FK)**	**Foreign key linking to user**
providerId	**INT (FK)**	**Foreign key linking to service provider**
ServiceId	**INT (FK)**	**Foreign key linking to Service**
subServiceId	**INT (FK)**	**Foreign key linking to subService**
status	**ENUM('Pending', 'Confirmed', 'Completed', 'Cancelled')**	**Booking status**
totalAmount	**DECIMAL(10,2)**	**Total service charge**
createdAt	**TIMESTAMP**	**Booking creation timestamp**

[Table 7.2.4**:** Services_Bookings**]**

Table 7.2.5: PAYMENTS

Description: This table stores information about task assignments to employees.

	PAYMENTS	
Type	**Type**	**Description**
INT (PK)	**INT (PK)**	**Unique payment ID**
INT (FK)	**INT (FK)**	**Foreign key linking to booking**
INT (FK)	**INT (FK)**	**Foreign key linking to user**
ENUM('Cash', 'Card', 'UPI')	**ENUM('Cash', 'Card', 'UPI')**	**Payment method used**
ENUM('Paid', 'Pending', 'Failed')	**ENUM('Paid', 'Pending', 'Failed')**	**Payment status**
VARCHAR(100)	**VARCHAR(100)**	**Unique transaction identifier**
TIMESTAMP	**TIMESTAMP**	**Payment timestamp**
VARCHAR(100)	**VARCHAR(100)**	**User Can Monitor the payment status and details**

[Table 7.2.5: Payment]

Table 7.2.6: BOOKINGS

Description: This table stores information about recent activities performed within the system**.**

	BOOKINGS	
Field Name	**Type**	**Description**
bookingId	**INT (PK)**	**Unique booking identifier**
userId	**INT (FK)**	**Foreign key linking to user**
providerId	**INT (FK)**	**Foreign key linking to service provider**
ServiceId	**INT (FK)**	**Foreign key linking to Service**
subServiceId	**INT (FK)**	**Foreign key linking to subService**
status	**ENUM('Pending', 'Confirmed', 'Completed', 'Cancelled')**	**Booking status**
totalAmount	**DECIMAL(10,2)**	**Total service charge**
createdAt	**TIMESTAMP**	**Booking creation timestamp**

[Table 7.2.6: BOOKINGS**]**

[Note: Many of the tables are yet to be created in the Data Dictionary. Further updates will include the necessary table structures to ensure comprehensive database documentation.**]**

CHAPTER:8

TESTING

8.1 Testing Plan

8.2 Testing Strategies

8.3 Testing Method

8.4 Test Case

8.1 Testing Plan

Great! Let's outline a detailed testing plan. Since you haven't specified the exact product or system you're testing, I'll provide a generic outline that you can adapt to your specific needs. Here's a structured testing plan:

- **Introduction:**
  - Provide an overview of the testing plan.
  - Include the objectives, scope, and key stakeholders.
- **Purpose:**
  - Define the purpose of testing.
  - Clarify what aspects of the product/system will be tested.
- **Scope:**
  - Describe the boundaries of the testing.
  - Identify the features, functionalities, or components included in the scope.
  - Mention any excluded items or functionalities.
- **Testing Goals:**
  - List the specific goals or objectives of the testing.
  - These goals should align with the overall project objectives and stakeholder requirements.
- **Testing Strategy:**
  - Outline the overall testing approach.
  - Specify the types of testing to be conducted (e.g., functional testing, usability testing, performance testing, security testing).
  - Describe any testing methodologies or frameworks to be followed (e.g., Agile, Waterfall, DevOps).
- **Test Deliverables:**
  - List the documents, reports, or artifacts that will be produced during testing.
  - Include items such as test plans, test cases, test scripts, and test reports.
- **Testing Environment:**
  - Specify the hardware, software, and network configurations required for testing.

- **Test Cases:**
  - Develop detailed test cases for each identified test scenario.
  - Include preconditions, steps to execute, expected results, and post-conditions.
  - Organize test cases by priority and complexity.
- **Test Execution:**
  - Describe how test cases will be executed.
  - Assign responsibilities for test execution.
  - Define the criteria for determining when testing is complete.
- **Defect Management:**
  - Explain how defects/issues will be logged, tracked, and managed.
  - Define severity and priority levels for defects.
  - Outline the defect resolution process.
- **Risks and Mitigation:**
  - Identify potential risks associated with testing.
  - Propose mitigation strategies for each identified risk.
- **Schedule:**
  - Create a testing timeline or schedule.
  - Allocate time for test preparation, execution, and reporting.
  - Consider any dependencies or constraints that may impact the schedule.
- **Roles and Responsibilities:**
  - Define the roles and responsibilities of team members involved in testing.
  - Specify who is responsible for test planning, execution, documentation, and communication.
- **Communication Plan:**
  - Outline how communication will be managed throughout the testing process.
  - Identify key stakeholders and their communication preferences.
  - Specify the frequency and format of status updates and reports.
- **Review and Approval:**
  - Establish a process for reviewing and approving the testing plan.
  - Identify the stakeholders responsible for review and approval.

- **Documentation:**
  - Document any changes or updates to the testing plan.
  - Maintain a version history for reference.
- **Conclusion:**
  - Summarize the key points of the testing plan.
  - Confirm alignment with project goals and stakeholder expectations.
- **Appendices:**
  - Include any supplementary information or reference materials relevant to the testing plan.

Once you've drafted your testing plan, be sure to review it with your team and stakeholders to ensure clarity, alignment, and agreement before proceeding with testing activities.

8.2 Testing Strategies

Testing strategies are crucial for ensuring that the testing process is efficient, thorough, and effective. Here's a detailed breakdown of various testing strategies:

- **Unit Testing:**
  - Definition: This strategy involves testing individual components or units of code in isolation.
  - Focus: It verifies the correctness of each unit's behavior and functionality.
  - Approach: Developers typically write unit tests using testing frameworks like JUnit (for Java), NUnit (for .NET), or pytest (for Python).
  - Benefits: Unit testing helps identify defects early in the development cycle and promotes code modularity and reusability.
- **Integration Testing:**
  - Definition: Integration testing verifies the interactions between different units or modules within the system.
  - Focus: It ensures that integrated components work together as expected.
  - Approach: Integration tests can be conducted using techniques such as top-down, bottom-up, or a combination of both.
  - Benefits: Integration testing detects interface defects and integration issues early, reducing the risk of major failures during system testing.

- **System Testing:**
    - Definition: System testing evaluates the entire system as a whole, including all integrated components.
    - Focus: It validates the system against its functional and non-functional requirements.
    - Approach: System testing can include functional testing, usability testing, performance testing, security testing, and more.
    - Benefits: System testing ensures that the software meets user expectations and performs reliably in its intended environment.
- **Acceptance Testing:**
    - Definition: Acceptance testing determines whether the system meets the acceptance criteria defined by stakeholders.
    - Focus: It validates the system from the end-user's perspective and assesses its readiness for deployment.
    - Approach: Acceptance testing can be performed using techniques such as alpha testing, beta testing, or user acceptance testing (UAT).
    - Benefits: Acceptance testing ensures that the software satisfies business requirements and aligns with user needs.
- **Regression Testing:**
    - Definition: Regression testing verifies that recent code changes have not adversely affected existing functionality.
    - Focus: It ensures that previously tested features still work correctly after modifications or enhancements.
    - Approach: Automated regression testing tools can help efficiently re-run existing test cases to detect regressions.
    - Benefits: Regression testing helps maintain software quality and prevents the introduction of unintended defects during development.
- **Exploratory Testing:**
    - Definition: Exploratory testing involves simultaneous learning, test design, and test execution.
    - Focus: Testers explore the application dynamically, uncovering defects and potential use cases.

- Approach: Testers rely on their domain knowledge and intuition to design and execute tests on-the-fly.
- Benefits: Exploratory testing is effective for finding defects that may not be identified through scripted tests and encourages creative and flexible testing approaches.

- **Risk-Based Testing:**
  - Definition: Risk-based testing prioritizes testing efforts based on the likelihood and impact of potential failures.
  - Focus: It focuses testing resources on high-risk areas of the system.
  - Approach: Risk analysis techniques, such as Failure Mode and Effects Analysis (FMEA), help identify critical areas for testing.
  - Benefits: Risk-based testing optimizes testing efforts by directing resources where they are most needed, reducing the likelihood of high-impact failures.
- **Load Testing:**
  - Definition: Load testing evaluates the system's performance under expected and peak load conditions.
  - Focus: It assesses how the system handles concurrent user activity and data processing.
  - Approach: Load testing tools simulate high-volume traffic to measure system response times, throughput, and scalability.
  - Benefits: Load testing identifies performance bottlenecks, helps optimize resource utilization, and ensures the system can handle expected workloads.
- **Security Testing:**
  - Definition: Security testing identifies vulnerabilities and weaknesses in the system's security controls.
  - Focus: It evaluates the system's ability to protect data, prevent unauthorized access, and resist attacks.
  - Approach: Security testing includes techniques such as penetration testing, vulnerability scanning, and code reviews.
  - Benefits: Security testing helps mitigate security risks, safeguard sensitive information, and maintain the integrity of the system.

- **Usability Testing:**
    - Definition: Usability testing assesses the system's user interface and user experience (UI/UX).
    - Focus: It evaluates how easily users can interact with the system and accomplish their tasks.
    - Approach: Usability testing involves observing real users as they interact with the system and collecting feedback through surveys or interviews.
    - Benefits: Usability testing identifies usability issues, improves user satisfaction, and enhances the overall user experience.

Each of these testing strategies plays a critical role in ensuring the quality, reliability, and performance of software systems. The selection and combination of testing strategies depend on factors such as project requirements, constraints, and risks. It's essential to tailor the testing approach to the specific needs of the project and continuously adapt it throughout the software development lifecycle.

8.3 Testing Method

Testing methods refer to the specific approaches or techniques used to conduct testing activities. These methods help ensure thorough coverage, accuracy, and effectiveness in validating the software's functionality, performance, and other quality attributes. Here are some common testing methods:

- **Black Box Testing:**
    - Description: In black box testing, the tester does not have access to the internal code or logic of the software. Instead, tests are designed based on the software's specifications and requirements.
    - Approach: Test cases are created to validate inputs, outputs, and system behavior without knowledge of the internal implementation.
    - Benefits: Black box testing focuses on the software's external behavior, allowing testers to identify issues from a user's perspective.
- **White Box Testing:**
- Description: White box testing, also known as glass box or clear box testing, involves examining the internal structure and logic of the software.

- Approach: Test cases are designed based on an understanding of the code, control flow, and data flow within the software.
- Benefits: White box testing provides insights into code coverage, control flow, and error handling, helping identify issues related to code execution and logic.

- **Gray Box Testing:**
    - Description: Gray box testing combines elements of both black box and white box testing. Testers have partial knowledge of the internal workings of the software.
    - Approach: Test cases are designed based on a limited understanding of the internal code or structure, allowing testers to target specific areas of interest.
    - Benefits: Gray box testing leverages both external and internal perspectives, providing a balance between test coverage and depth.
- **Manual Testing:**
    - Description: Manual testing involves human testers executing test cases manually without the use of automation tools.
    - Approach: Testers interact with the software interface, input data, and observe outputs to validate its behavior.
    - Benefits: Manual testing allows for exploratory testing, subjective evaluation of user experience, and validation of complex scenarios that may be challenging to automate.
- **Automated Testing:**
    - Description: Automated testing involves the use of software tools to execute pre- scripted tests, compare actual outcomes with expected results, and generate test reports.
    - Approach: Test scripts are created to automate repetitive test scenarios, regression tests, and performance tests.
    - Benefits: Automated testing improves efficiency, repeatability, and coverage of testing activities, especially for large-scale projects and continuous integration pipelines.

- **Static Testing:**
    - Description: Static testing focuses on reviewing code, requirements, and documentation without executing the software.
    - Approach: Techniques such as code reviews, walkthroughs, and inspections are used to identify defects and improve quality early in the development lifecycle.
    - Benefits: Static testing helps uncover defects at an early stage, reducing the cost and effort of fixing issues later in the development process.

- **Dynamic Testing:**
    - Description: Dynamic testing involves executing the software and observing its behavior to validate its functionality, performance, and other quality attributes.
    - Approach: Techniques such as unit testing, integration testing, system testing, and performance testing fall under dynamic testing.
    - Benefits: Dynamic testing verifies the software's behavior in a real-world environment, uncovering defects that may not be apparent through static analysis alone.
- **Model-Based Testing:**
    - Description: Model-based testing involves creating abstract models of the system's behavior and using them to generate test cases automatically.
    - Approach: Models, such as finite state machines or UML diagrams, are analyzed to derive test scenarios and expected outcomes.
    - Benefits: Model-based testing improves test coverage, reduces manual effort in test case design, and ensures consistency between requirements and tests.

These testing methods can be combined and adapted based on project requirements, constraints, and goals. Effective testing often involves a combination of multiple methods to achieve comprehensive coverage and ensure software quality.

8.4 Test Case

A test case is a detailed set of conditions or steps that are designed to verify the functionality, behavior, or performance of a software application or system. Test cases are used to validate whether the software meets specific requirements and functions correctly under various scenarios. Here's a breakdown of what a test case typically includes:

- **Test Case ID:** A unique identifier for the test case, usually in the form of a combination of letters and numbers.

- **Test Case Title/Name:** A descriptive title or name that summarizes the purpose or objective of the test case.
- **Description:** A brief description or summary of what the test case is intended to achieve.
- **Preconditions:** Any prerequisites or conditions that must be met before executing the test case, such as system configurations, data setup, or user permissions.
- **Test Steps:** A sequence of step-by-step instructions outlining the actions to be performed to execute the test case. Each step should be clear, concise, and unambiguous.
- **Input Data:** The specific data values, parameters, or inputs that will be used during the test execution.
- **Expected Results:** The expected outcomes or behaviors that the system should exhibit when the test case is executed successfully. This includes both observable results and system responses.
- **Actual Results:** The actual outcomes observed during test execution. Testers record the results obtained after executing the test steps.
- **Pass/Fail Criteria:** Criteria used to determine whether the test case passes or fails based on a comparison between the actual results and the expected results.
- **Priority:** The priority level assigned to the test case, indicating its relative importance or urgency in the testing process.
- **Severity:** The severity level assigned to the test case, indicating the impact of a failure on the system or its users. Common severity levels include critical, major, minor, and trivial.
- **Test Environment:** Information about the test environment, including hardware, software, configurations, and dependencies required for test execution.
- **Test Data:** Any additional test data or conditions relevant to the test case, such as boundary values, edge cases, or negative scenarios.
- **Attachments/References:** Any supporting documents, screenshots, logs, or references that provide additional context or details for the test case.
- **Notes/Comments:** Any additional notes, comments, or observations relevant to the test case, such as insights, assumptions, or recommendations.

By documenting test cases in a structured and standardized format, teams can ensure consistent testing practices, facilitate test execution, and maintain comprehensive test coverage throughout the software development lifecycle. Additionally, well-written test cases serve as valuable documentation for future reference, regression testing, and knowledge transfer within the team.

CHAPTER: 9

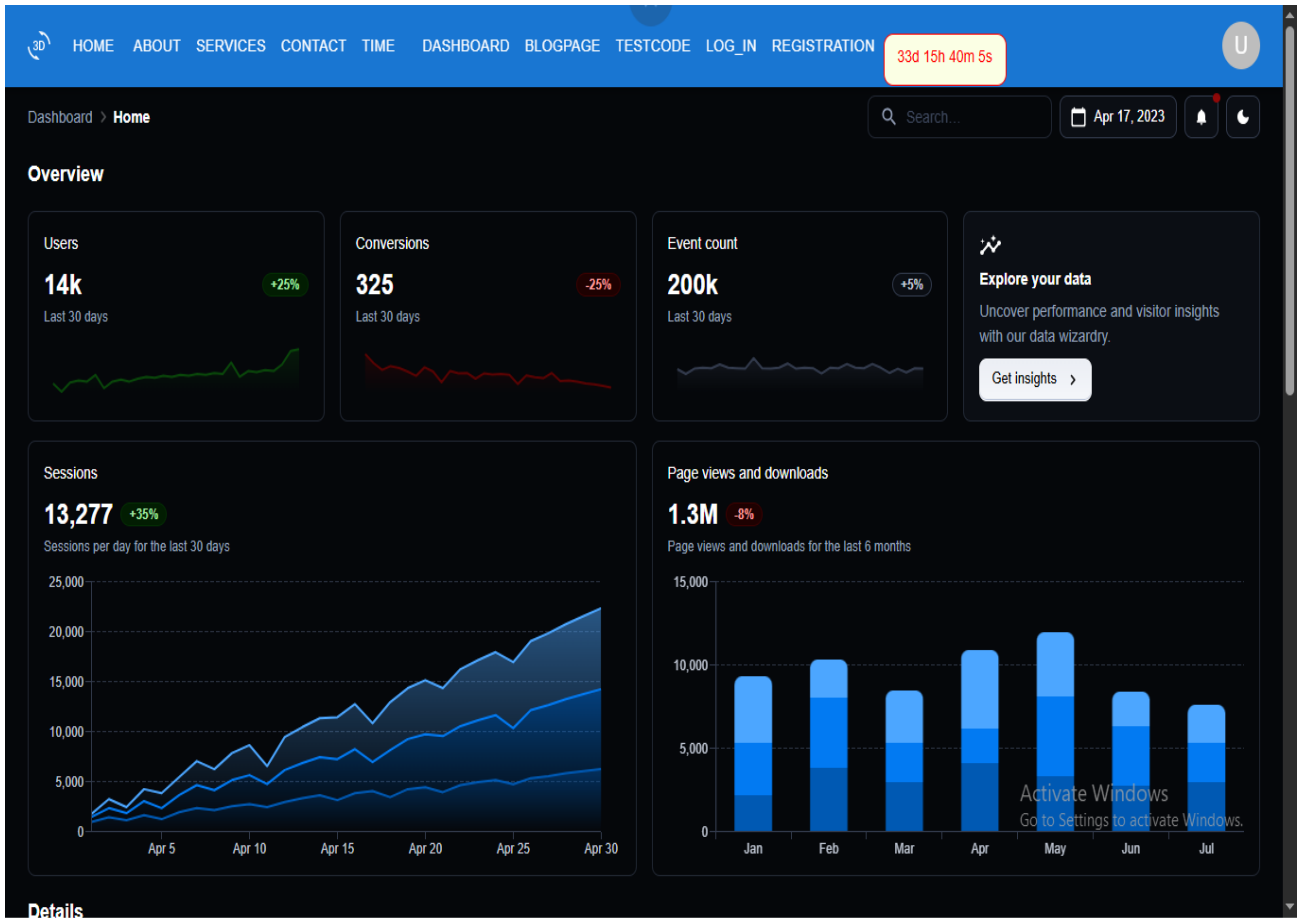
SNAPSHOT OF WEBSITE

9.1 Admin Dash Board

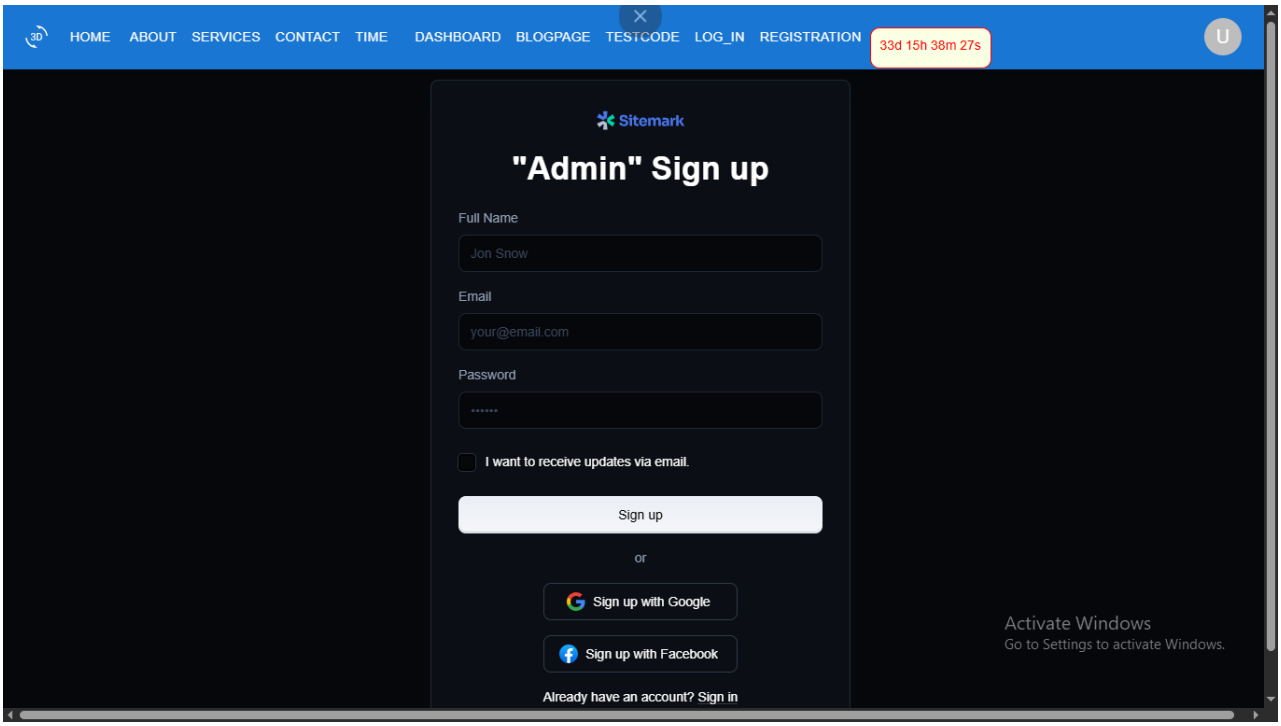
9.2 User Admin and Provider Login

9.3 User Admin and Provider Registration

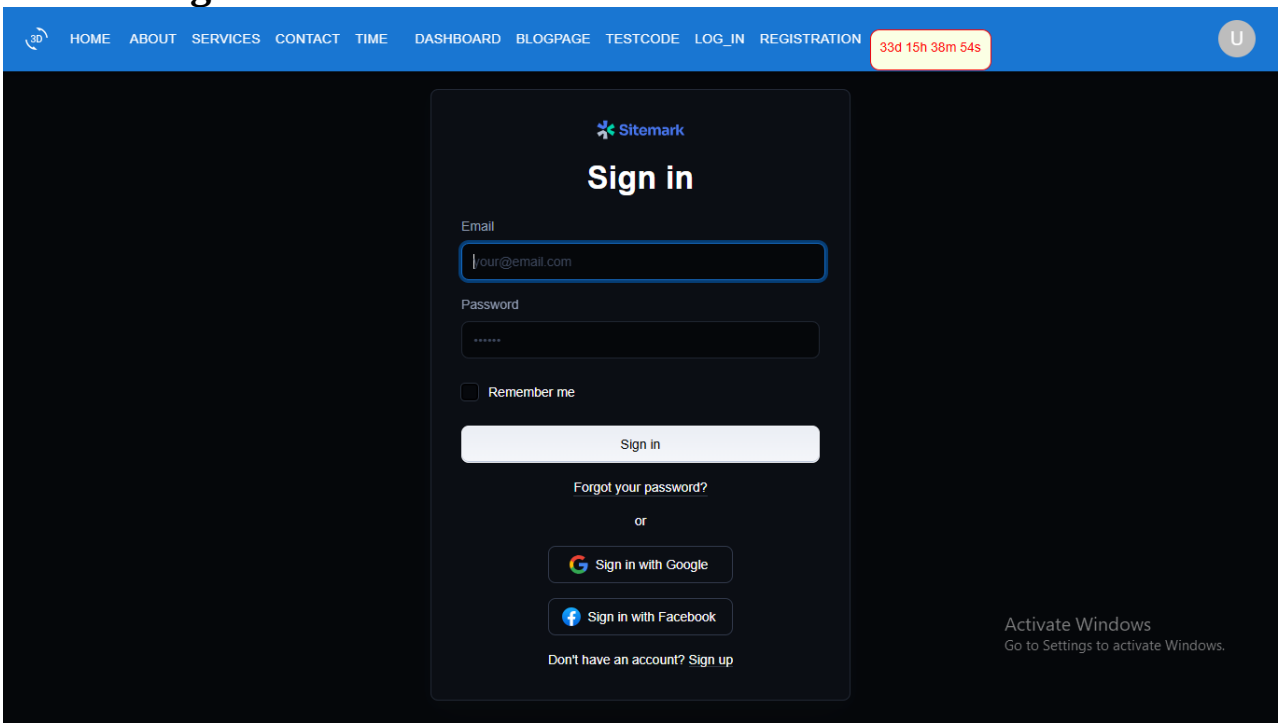
Admin Dash Board:



Admin Sign Up:



Admin Log In



:

CHAPTER: 10

ADVANTAGE

10.1 Advantages

10.2 Limitations

10.1 Advantages

- **Efficiency:** It enhances productivity by efficiently allocating tasks to resources based on their availability, skills, and priority.
- **Time Management:** It helps in effective time management by providing a clear schedule of tasks, deadlines, and milestones, thereby reducing time wasted on indecision or confusion.
- **Resource Optimization:** By matching tasks with available resources, it ensures that resources are utilized optimally, reducing idle time and maximizing output.
- **Prioritization:** It enables users to prioritize tasks based on their importance and urgency, ensuring that critical tasks are completed on time.
- **Coordination:** Facilitates better coordination among team members by providing visibility into everyone's tasks, schedules, and progress, thus reducing conflicts and improving collaboration.
- **Real-time Updates:** Provides real-time updates on task progress, allowing managers to track project status, identify bottlenecks, and make timely adjustments.
- **Data-driven Insights:** Generates valuable data and insights about task completion rates, resource utilization, and project timelines, which can be used for performance evaluation, process improvement, and future planning.
- **Reduced Errors:** Minimizes errors and oversights by automating task assignment and scheduling processes, leading to smoother operations and higher-quality outcomes.
- **Adaptability:** Offers flexibility to adapt to changing priorities, deadlines, and resource availability, ensuring that schedules remain dynamic and responsive to evolving circumstances.
- **Cost Savings:** By improving efficiency, reducing downtime, and preventing delays, it helps in controlling costs associated with project management and resource allocation.

10.2 Limitations

Task Monitoring & Scheduling systems offer significant benefits, they also have some limitations:

- **Dependency on Data Accuracy:** The effectiveness of these systems relies heavily on accurate input data regarding task details, resource availability, and dependencies. Inaccurate or incomplete data can lead to scheduling errors and inefficiencies.
- **Complexity:** Implementing and managing a sophisticated task monitoring and scheduling system can be complex, requiring training for users and administrators. Complexity can sometimes lead to resistance to adoption or errors in system operation.
- **Integration Challenges:** Integrating the task monitoring and scheduling system with existing software and workflows can be challenging, especially in organizations with diverse systems and platforms. Compatibility issues may arise, requiring additional resources for integration efforts.
- **Scalability Issues:** As organizations grow or project complexity increases, scalability can become a challenge for task monitoring and scheduling systems. Ensuring that the system can accommodate larger datasets, more users, and additional features is essential for long-term viability.
- **Security Concerns:** Storing sensitive task-related data in a centralized system poses security risks, including data breaches, unauthorized access, or data loss. Implementing robust security measures is crucial to protect sensitive information.
- **Maintenance and Updates:** Ongoing maintenance, updates, and support are necessary to keep the task monitoring and scheduling system running smoothly and up-to-date with evolving requirements and technology trends. This requires dedicated resources and commitment from the organization.

CHAPTER: 11

CONCLUSION AND FUTURE ENHANCEMENT

11.1 Conclusion

11.2 Future Enhancement

11.1 Conclusion

In conclusion, a Task Monitoring & Scheduling system offers a multitude of benefits, including increased efficiency, improved time management, optimized resource utilization, better coordination, and valuable insights for decision-making. However, it's important to acknowledge the limitations associated with these systems, such as dependency on data accuracy, complexity, initial investment costs, integration challenges, and human factors.

Task Monitoring & Scheduling system outweigh the challenges, particularly in today's fast-paced and competitive business environment. By leveraging technology to automate and streamline task management processes, organizations can enhance productivity, reduce errors, and improve overall project outcomes.

To maximize the effectiveness of a Task Monitoring & Scheduling system, it's crucial to address potential challenges through proper planning, training, integration efforts, and ongoing maintenance. Additionally, fostering a culture of adaptability and open communication can help overcome resistance to change and ensure successful adoption of the system across the organization.

In essence, while no system is without its limitations, a well-designed and properly implemented Task Monitoring & Scheduling system can significantly enhance organizational efficiency, effectiveness, and competitiveness in today's dynamic business landscape.

11.2 Future Enhancement

Future enhancements to Task Monitoring & Scheduling systems could focus on addressing existing limitations and incorporating emerging technologies to further improve efficiency, effectiveness, and user experience.

Here are some potential areas for enhancement:

- **AI and Machine Learning Integration:** Implementing AI and machine learning algorithms can enhance task scheduling by predicting resource availability, identifying patterns in task completion times, and recommending optimized schedules based on historical data.

- **Advanced Analytics:** Enhancing analytical capabilities to provide deeper insights into task performance, resource utilization, and project trends can enable more informed decision-making and proactive management of tasks and schedules.
- **Predictive Analytics for Risk Management:** Integrate predictive analytics capabilities to identify potential risks and anticipate delays in task completion, allowing for proactive risk mitigation and adjustment of schedules to minimize impacts on project timelines.
- **Mobile Accessibility:** Improve mobile accessibility to allow users to monitor tasks, update statuses, and view schedules on-the-go, enhancing flexibility and enabling remote work scenarios.
- **Integration with IoT Devices:** Integrate with IoT devices and sensors to automatically capture task progress and resource utilization data, providing real-time insights and enabling more accurate scheduling decisions.
- **Dynamic Resource Allocation:** Develop algorithms for dynamic resource allocation, allowing the system to automatically adjust task assignments and schedules in response to changing priorities, resource availability, and project requirements.
- **Customization and Flexibility:** Provide greater customization options and flexibility in configuring task monitoring and scheduling rules, workflows, and user interfaces to accommodate diverse organizational needs and preferences.

By continuously innovating and enhancing Task Monitoring & Scheduling systems in these areas, organizations can stay ahead of the curve, drive operational excellence, and achieve greater success in managing tasks, projects, and resources effectively.

BIBLIOGRAPHY

a. **Course Outcome**

b. **Books**

c. **Web Reference**

a. Course Outcome

Sr. No.	CO statement	Marks % weightage	Chapter No.
CO-1	Undertake problem identification, formulation and solution	20%	**CHAPTER 1** INTRODUCTION
CO-2	Design engineering solutions to complex problems utilising a systematic approach and team work	30%	**CHAPTER 2** SYSTEM REQUIREMENTS **CHAPTER 3** DAILY TASK
CO-3	Communicate with engineers and the community at large in written and oral forms	20%	**CHAPTER 6** SYSTEM DESIGN **CHAPTER 10** ADVANTAGES **CHAPTER 11** CONCLUSION AND FUTURE ENHANCEMENT
CO-4	Demonstrate the knowledge and understanding of engineering and management principle and apply it to assigned project	30%	**CHAPTER 4** FRONT END OF SYSTEM **CHAPTER 5** BACK END OF SYSTEM **CHAPTER 6** SYSTEM DESIGN **CHAPTER 7** DATA DICTIONARY **CHAPTER 8** TESTING

b. Books:

- **Fundamental of Software Engineering**
  - Author: Rajib Mall
- **The Joy of PHP Programming: A Beginner's Guide to Programming Interactive Web Applications with PHP and Mongo DB**
  - Author: Alan Forbes
- **Head First PHP & Mongo DB**
  - Author: Lynn Beighley & Michael Morrison
- **SQL Server Database**
  - Author: Mr.Gaurav Desani

c. Web Reference:

- **W3Schools. "CSS Tutorial." [Online].**
  - Available: https://www.w3schools.com/css/default.asp.. [Accessed: February 1, 2025].
- **W3Schools. "JavaScript Tutorial." [Online].**
  - Available: https://www.w3schools.com/js/default.asp. [Accessed: February 7, 2025].
- **Javatpoint. "PHP Tutorial."**
  - [Online] Available: https://www.javatpoint.com/php-tutorial [Accessed: February 21, 2025].
- **Free Code Camp. "Learn to code — for free." [Online].**
  - Available: https://www.freecodecamp.org/ [Accessed: March 15, 2025].
- **Stack Overflow. "Where Developers Learn, Share, & Build Careers." [Online].**
  - Available: https://stackoverflow.com/ [Accessed: April 2, 2025]