Practical No. 1

Study of Web and its basics

Problem Statement 1:

Study and describe the following concepts in your words:

1. Evolution of Web including web 3.0

2. Which ports and protocols are used by web? Describe those in detail

3. Difference Between HTTP & HTTPS

4. What is web developer tools and why it is needed?

5. Elaborate with diagram client server architecture and MVC architecture. When to use which architecture?

6. What is HTML and HTML5?

7. Which are the currently used versions of HTML and CSS for web development?

8. Which tools are available for front end development and back end development?

9. What MERN stack includes? Why and when it is preferred for web development?

10. List out newly introduced input types, APIs, form elements, and elements that support media content in HTML5.

11. Explain HTML5 Web storage.

Problem Statement 2:

Study of Different HTML and CSS tags:

1. Study different tags of HTML and CSS

2. Create a static web page for "Portfolio" of your own. Which will include photo, name, class, College name, Achievements/ Certificates, Extracurricular Activities, Courses Completed, hobbies, Technical expertise, etc

Note:

1. Create a **document** of the above website with screenshots.
2. Scan the document and **create a pdf file** with **"ExamSeatNum_P#PS#" as its name**.
3. Upload the file on the **WCE Moodle** before the given deadline.

Practical No. 2

Study of HTML and CSS.

Create a website which displays the course information using static web pages

Problem Statement 1: Basic HTML

- Create a HTML file named "course.html" with a proper document structure.
- Add a title to the webpage: "Introduction to [Your Course Name] Course".
- Create a header section with the course name as the main heading and a brief description as a subheading.
- Design a navigation menu with links to different sections of the page: "About the Course," "Course Outline," and "Instructors." (For this you may create separate pages and link them or you can create it on single page. But when you click on the navigation menu only those particular contents will be displayed)

Problem Statement 2: Styling with CSS

- Create a separate CSS file named "styles.css" and link it to the "course.html" file.
- Style the header section with a background colour, white text colour, and proper padding.
- Apply a font-family of your choice to the entire document.
- Add a background colour to the navigation menu items when hovered over.
- Style the course description text with a distinctive font style and colour.
- Create a two-column layout for the "About the Course" and "Course Outline" sections using CSS Grid or Flexbox.
- Apply a box-shadow to the instructor cards in the "Instructors" section.
- Style the footer with a dark background colour, light text colour, and centred content.

Problem Statement 3: Course Content Display

- In the "About the Course" section, provide a detailed description of the course, including its objectives and target audience.
- Create an ordered or unordered list in the "Course Outline" section, outlining at least Six modules/topics covered in the course.
- Display instructor information in the "Instructors" section, including their names, photos, and a brief bio.

Problem Statement 4: Responsive Design

- Implement responsive design techniques to make the website mobile-friendly.
- Ensure that the navigation menu collapses into a hamburger menu on smaller screens.
- Adjust the layout and font sizes for optimal viewing on various devices.

Problem Statement 5: Extra Enhancements

- Create a hover effect for the instructor cards, such as changing the border colour or adding a shadow.
- Add transition effects to navigation menu items and buttons.
- Implement a Google Fonts integration to style the text fonts.

Other than the above problem statements you may add some Interactive Elements as per your choice. Some of them are as follows:

- Add a button in the "Course Outline" section that, when clicked, reveals additional details about each module.
- Create a "Contact Us" section at the end of the page with a form for users to submit their inquiries.

Note:

1. Create a **document** of the above website with screenshots.
2. Scan the document and **create a pdf file** with **"ExamSeatNum_P#PS#" as its name**.
3. Upload the file on the **WCE Moodle** before the given deadline.

Practical No. 3

Study of Java Script

Problem Statement 1: Basics of Java Script

Write a JavaScript function that takes an array of numbers as input and returns the sum of all even numbers in the array. Additionally, create a simple HTML interface with an input field where the user can enter comma-separated numbers and a button to trigger the calculation. Display the result below the button.

Problem Statement 2: Implement a function that validates a password based on the following criteria:

- At least 8 characters long.
- Contains at least one uppercase letter, one lowercase letter, one digit, and one special character.

Create an HTML form with an input field for the password and a "Check Password" button. Display a message indicating whether the password is valid or not.

Problem Statement 3: Web Development

Create a basic web page for a fictional online bookstore using HTML, CSS, and JavaScript. The web page should have the following features:

- A header with the bookstore name and a navigation menu (Home, Books, Contact).
- An area to showcase featured books with their titles, authors, and cover images.
- A section where all available books are displayed in a grid format. Each book should have an image, title, author, and a button to "Add to Cart".
- Implement a simple shopping cart functionality using JavaScript. Allow users to add books to their cart and display the total number of items in the cart on the navigation menu.

Problem Statement 4: Responsive Design

- Implement responsive design techniques to make the website mobile-friendly.
- Ensure that the navigation menu collapses into a hamburger menu on smaller screens.
- Adjust the layout and font sizes for optimal viewing on various devices.

IMP: for problem statement 1 and 3 different groups should take different problems like for example for problem statement 1 instead of returning sum of all even numbers one might return sum of all odd numbers, any other mathematical operations, etc.

Also for problem statement you should create different websites, like online food ordering system or online grocery shop, etc.

Every group should have a different application. These are the basic guidelines provided to you.

Note:

1. Create a **document** of the above website with screenshots.
2. Scan the document and **create a pdf file** with **"ExamSeatNum_P#PS#" as its name**.
3. Upload the file on the **WCE Moodle/ERP** before the given deadline.

Practical No. 4

Study of Java Script and DOM.

Perform following problem statements for DOM using Javascript

Problem Statement 0: Basics of DOM

- What is the DOM?
- What is DOM Tree Structure? Elaborate its elements with example (you may create DOM tree structure of previously created web pages)
- Give examples for following:
  - Accessing the DOM
  - Manipulating the DOM
  - Event Handling
  - Traversing the DOM
- What are Performance Considerations while implementing the DOM and can DOM supports all browsers?
- Elaborate Common Methods and Properties of DOM

Problem Statement 1: DOM selector methods

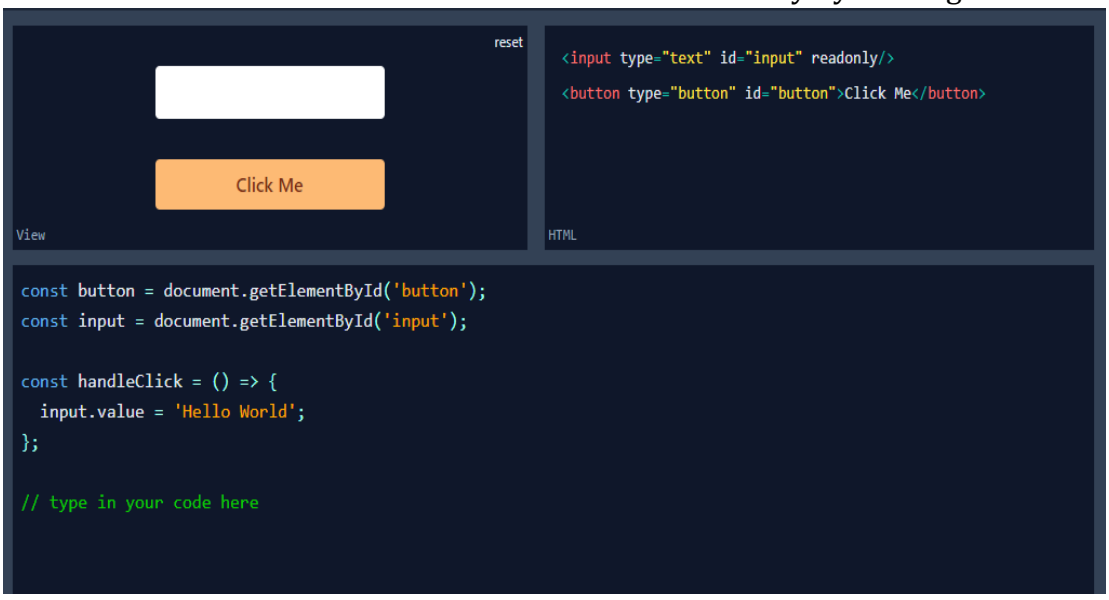
- Here, the existing code expects the variables 'buttonElem' and 'inputElem' to represent the button and input elements in the example UI. Assign the respective elements to the variables. In this case, the two elements do not have unique identifiers - like for example an id. Instead they are direct descendents of a div element with id 'wrapper'. Use an appropriate selector method! Click the button to verify that the code is working.

- In this scenario, we are looking for a list of elements gathered in one variable - rather than only one element. Assign the list items in the view to the variable 'listItems' by using an appropriate selector method. Once you have completed the

code below, verify it by hovering over the list items until all items have the value 'ON'

Problem Statement 2: Events and user interactions

- The Javascript function handleText fills the input field with the words Hello World. But, there is no code to execute this function. Complete the existing code below such that the function is called when the button is clicked. Verify by clicking the button.



- The Javascript function changeText changes the text inside the circle. But again, there is no code to execute this function. Complete the existing code below such that the function is called when the cursor moves onto the circle. Verify that your code works by hovering over the circle.

- In this scenario we want the color of the circle to change depending on the type of cursor movement. Use the function toggleColor to turn the circle orange when the cursor moves onto it. Reuse the same function to turn it black when the cursor leaves it. The tricky part is that you have to call toggleColor with different values for the parameter isEntering. Verify that your code is working by hovering the circle with the mouse cursor and leaving it again.

Problem Statement 3: DOM manipulation with JavaScript

- Remove element from the DOM. Create t2 circles red and green and a button clickme. Place them such a way that red circle hides the green circle. Add the function removeRedCircle to remove the circle with id red from the DOM when clicked on clickme button. Make sure that you really remove the element instead of just hiding it.

Problem Statement 4: DOM fundamentals

- Create JavaScript code to interact with the displayed HTML elements. Create a checkbox and a button. Once you click the button, the checkbox should be checked.
- Create 3 textboxes and a button. First 2 checkboxes contain first name and last name respectively. When the button is clicked, combine the names of the first two input fields. Insert the full name in the third input field (textbox). Check if your code still works if you change the first or last name.
- Create three buttons. One button displays value of 0. Other two buttons are for increment and reset. By clicking increment button each time, increase the value of the button by 1. By clicking the reset button set the value of button to 0. Confirm your code by clicking the buttons.
- create a dynamic input filter with JavaScript. Type a search term in the input field. The displayed items in the list should match your search term. The rest of the list elements should be hidden.
- Create 10 balloons as shown below. Every time you hover over a balloon, it should become invisible. Your goal is to pop all the balloons one after the other. Create a refresh button. After clicking refresh button it will again display all the balloons.

Problem Statement 5: Recursive functions

- Create a function move that moves the button 1px to the left or the right. It is recursive because it calls itself again and again. This keeps the button moving. Extend the JavaScript code. Once you click the button, it should stop moving. When you click it again, it should move again.

Note:

1. Create a **document** of the above website with screenshots.
2. Scan the document and **create a pdf file** with **"ExamSeatNum_P#PS#" as its name**.
3. Upload the file on the **WCE Moodle** before the given deadline.

Practical No. 5

Study and implementation of ReactJs

Perform following problem statements using ReactJs

Problem Statement 0: Basics of ReactJs

- What is React and what problem does it solve?
- What are React components and how are they used?
- What is JSX in React?
- What are props in React and how do they differ from state?
- What is state in React and how does it work?
- What are React lifecycle methods, and why are they important?
- Elaborate following with respect to ReactJs
    - Event Handling
    - Conditional Rendering
    - Lists and Keys
    - Forms
    - Hooks
    - React Router
    - State Management
    - React Context API
- How can you optimize the performance of a React application?

Problem Statement 1: Star Wars character app

(In this problem statement, example of Star Wars is given, you may choose any characters from the series of the movie like Harry Potter, etc. Every group in a batch will have different characters.)

- Using a public API, display a list of all Star Wars characters using the endpoint "/people". The API has paging, so the developer must also implement pagination. Also, a simple loader for fetching/refetching data as well as handling the error state (i.e., if the API server is down).
- For every user, we'd like to display a card with the name of each character along with a random picture for each character (see Picsum photos for random picture inspiration). Each character card should be colored based on their species and have some kind of animation when the user hovers over the card. When we click on a character's card, more information should appear in a modal about the character.
- In the character details modal, we'd like to display information about the person: name as the header of the modal, height displayed in meters, mass in kg, date

person was added to the API (in dd-MM-yyyy format), number of films the person appears in and their birth year. We should also fetch information about the person's homeworld and display its name, terrain, climate, and amount of residents.

Note:

1. Create a **document** of the above website with screenshots.
2. Scan the document and **create a pdf file** with **"ExamSeatNum_P#PS#" as its name**.
3. Upload the file on the **WCE Moodle** before the given deadline.

Practical No. 6

Study and implementation of ReactJs

Perform following problem statements using ReactJs

Problem Statement 1: To-do List Application with State Management

- Build a simple To-do List application that allows users to add, remove, and mark tasks as completed.
- The app should have a responsive user interface that works well on both desktop and mobile devices.
- Users should be able to input a task, which will be added to the list.
- Each task should have a checkbox to mark it as completed.
- Users can remove tasks from the list.
- Use React hooks like *useState* for state management.
- Add options to filter tasks based on completion status (e.g., all, completed, active).
- Break down the Todo List into multiple components, like TaskList, TaskItem, AddTaskForm, etc.
- The app should display appropriate error messages.
- The candidate should use appropriate error handling and validation to ensure that the application is robust and user-friendly.

Problem Statement 2: Simple E-commerce Cart System

Requirements

- Develop a small e-commerce application where users can browse products and add them to a shopping cart.
- Display a list of products with details like price, name, and image.
- Users can add products to a shopping cart.
- The cart page shows all selected products and their total price.
- Users can remove items from the cart or adjust the quantity.
- Add filtering options (e.g., by price, category) and sorting (e.g., low-to-high, high-to-low).
- Add routing to navigate between different pages, like the product list, cart, and checkout pages.
- On clicking a product, navigate to a detailed view of that product.

Problem Statement 3: User Authentication with React Context API

Requirements

- Create a login form that checks for hardcoded user credentials.
- If authenticated, the user can see a personalized dashboard.
- Allow the user to log out, which resets the context state.
- Use React Router for navigation between login and dashboard pages.
- Protect certain routes (e.g., dashboard) so that only authenticated users can access them.

Note:

1. Create a **document** of the above website with screenshots.
2. Scan the document and **create a pdf file** with **"ExamSeatNum_P#PS#" as its name**.
3. Upload the file on the **WCE /ERP** before the given deadline.

Practical No. 7

Study and implementation of Express.js

Perform following problem statements using ReactJs

Problem Statement 1: Basics of Express.js

- What is Express.js and how does it differ from Node.js?
- How do you create a simple Express.js server?
- Explain the concept of routing in Express.js. How do you define routes?
- What is middleware in Express.js, and how does it work?
- How do you create and use custom middleware in an Express.js application?
- What is the difference between application-level middleware and router-level middleware?
- What are req and res in Express.js? Give examples of common properties and methods associated with each.
- How would you extract query parameters from a URL in an Express.js route?
- How does Express.js handle different HTTP methods (GET, POST, PUT, DELETE)?
- What are route parameters in Express.js? How do you use them in a route definition?

Problem Statement 2: Basic Web Server with Express.js

Requirements

- Create a basic Express.js server that listens on port 3000.
- Define three routes:
  - GET / - Responds with "Welcome to the Home Page".
  - GET /about - Responds with "This is the About Page".
  - GET /contact - Responds with "Contact us at: email@example.com".
- Include a 404 error handler that displays a "Page Not Found" message for unknown routes.

Problem Statement 3: Dynamic Route Parameters

Requirements

- Modify the previous server to include the following route:
- GET /users/:id - Responds with "User ID: [id]" where [id] is the dynamic value from the route.
- Add another route:
  - GET /products/:category/:productId - Responds with "Category: [category], Product ID: [productId]".

- Return a JSON object containing the category and product ID instead of a plain string.Note:

1. Create a **document** of the above website with screenshots.
2. Scan the document and **create a pdf file** with **"ExamSeatNum_P#PS#" as its name**.
3. Upload the file on the **WCE /ERP** before the given deadline.

Practical No. 8

Study and implementation of node.js

Perform following problem statements using Node.js

Problem Statement 1: Introduction to Node.js

- What is Node.js, and how does it differ from traditional server-side platforms like Apache or PHP?
- What is the purpose of the V8 engine in Node.js?
- Explain the single-threaded, event-driven architecture of Node.js.
- Why is Node.js considered non-blocking?
- What is npm, and how is it used in Node.js?
- What is a module in Node.js? How do you export and import modules?
- What is the difference between require() and import in Node.js?
- How can you create a custom module in Node.js?
- What is the role of the package.json file in a Node.js project?
- How do you install a package globally and locally using npm?
- What is the difference between asynchronous and synchronous programming in Node.js?
- How do you create an HTTP server in Node.js?
- What is the difference between http.createServer() and using frameworks like Express.js?
- How do you handle GET and POST requests in Node.js?

Problem Statement 2: Middleware (Express.js)

- What is middleware in Node.js, particularly in the context of Express.js?
- How do you create custom middleware in Express.js?
- Explain how middleware is executed in order in an Express.js application.

Problem Statement 3: File System (fs) Module

- How do you read and write files using the fs module in Node.js?
- What is the difference between fs.readFile() and fs.readFileSync()?
- How can you check if a file or directory exists in Node.js?
- How do you handle file operations in an asynchronous manner?

Problem Statement 4: Database Connectivity

- How do you connect to a SQL or Oracle database from a Node.js application?
- What is the purpose of the mysql2 library in Node.js?

- Explain how you would perform basic CRUD operations (Create, Read, Update, Delete) using MySQL and Node.js.

Problem Statement 5: Building a RESTful API

Develop a RESTful API using Node.js and Express.js for a library management system. The system should allow users to:

- Add new books (title, author, genre, year of publication).
- Update book details.
- Delete books from the collection.
- Fetch a list of books with pagination and filtering by genre and author.
- Add a user authentication system to restrict access to certain API routes.

File Upload and Management System

Build a file upload and management system using Node.js and Multer (or any other file upload middleware). The system should allow users to:

- Upload files (images, PDFs, etc.).
- View the list of uploaded files.
- Download or delete specific files.
- Implement user authentication so that only authorized users can upload and manage files.

1. Create a **document** of the above questions with screenshots wherever necessary.
2. Scan the document and **create a pdf file** with **"ExamSeatNum_P#PS#" as its name**.
3. Upload the file on the **WCE /ERP** before the given deadline.

Practical No. 9

Study and implementation of node.js

Perform following problem statements using Node.js

Problem Statement 1: Database Connectivity using SQL or Oracle

- Write a Node.js program that connects to an Oracle/SQL database, retrieves data from a table, and displays the results.

Problem Statement 2: Middleware (Express.js)

- What is middleware in Node.js, particularly in the context of Express.js?
- How do you create custom middleware in Express.js?
- Explain how middleware is executed in order in an Express.js application.

Problem Statement 3: File System (fs) Module

- How do you read and write files using the fs module in Node.js?
- What is the difference between fs.readFile() and fs.readFileSync()?
- How can you check if a file or directory exists in Node.js?
- How do you handle file operations in an asynchronous manner?

Problem Statement 4: File Upload and Download API

Develop a file upload and download API using Node.js and Express. The API should allow users to upload files (e.g., images, documents) and download them later.

- Create an API to upload files to the server.
- Implement routes to retrieve and download files.
- Ensure proper error handling (e.g., file size limits, invalid file formats).
- Implement file versioning to allow multiple uploads of the same file name without overwriting.

Problem Statement 5: Real-time Chat Application with Socket.io

Create a real-time chat application using Node.js, Express, and Socket.io that allows multiple users to join and communicate in a chat room.

- Set up a Node.js server with Socket.io for real-time bi-directional communication.
- Implement event listeners to handle user connections, disconnections, and message broadcasting to all connected users.

1. Create a **document** of the above questions.
2. Scan the document and **create a pdf file** with **"ExamSeatNum_P#PS#" as its name**.
3. Upload the file on the **WCE /ERP** before the given deadline.

Practical No. 10

Projects using MySQL, NodeJs and ExpressJs, Reacrjs

Perform following problem statements using Node.js

Problem Statement 1: Personal Task Manager

- Develop a simple task manager where users can add, edit, and delete their personal tasks. The tasks should have fields like title, description, dueDate, and status (pending/completed).
- User authentication (JWT-based login/signup).
- CRUD operations for tasks (Create, Read, Update, Delete).
- Filtering tasks by status (pending/completed).
- Allow users to categorize tasks and set priorities (high, medium, low).

Problem Statement 2: User Profile Management System

- Build a profile management system where users can create accounts and update their personal information such as name, email, phone number, and address.
- User registration and authentication (using JWT).
- Profile CRUD: Users can create, view, edit, and delete their profiles.
- Add the ability to upload profile pictures and store the file path in MySQL.

Problem Statement 3: Simple Blog Application

- Create a simple blog platform where users can sign up, log in, and write, edit, or delete their blog posts. Other users can view the blog posts but not edit or delete them.
- User authentication (JWT).
- Blog post CRUD operations.
- Display all blog posts on the homepage, with individual post details accessible through a unique URL.
- Add comment functionality where users can comment on blog posts.

Problem Statement 4: Online Contact Book

- Create an online contact book where users can store, update, and delete their contacts. Each contact has details like name, phone number, email, and address.
- User authentication.
- CRUD operations for contacts.
- Display contacts in a list with pagination.
- Add search functionality to search contacts by name or email.
- Import and export contacts as CSV files.

Problem Statement 5: Event Planner Application

- Create an event planner app where users can organize events, including event details such as event name, date, location, and description.
- User authentication and role-based access (e.g., admin and user).
- CRUD operations for events (only admins can create, edit, or delete events).
- Display event details for all users.
- Implement event registration functionality, where users can register for events.
- Add email notifications when users register for an event.

Implement any 2 problem statement from the above 5 problem statements

1. Create a **document** of the above questions.
2. Scan the document and **create a pdf file** with **"ExamSeatNum_P#PS#" as its name**.
3. Upload the file on the **WCE /ERP** before the given deadline.