Lab 01: Revision – Javascript

1. WAP in JavaScript to make simple calculator using popup box. (A)
2. WAP in JavaScript to check whether the given no. is prime or not. (A)
3. WAP in JavaScript to fnd the factorial of given number. (A)
4. WAP in JavaScript to print the Fibonacci series of a number. (A)
5. WAP in JavaScript to check whether the given number is palindrome or not. (A)
6. WAP in JavaScript to check whether the given number is Armstrong or not. (A)
7. WAP in JavaScript to print the factors of given number. (A)
8. WAP in JavaScript to prime number between the given range of number. (A)
9. WAP to display given patterns using JavaScript. (A)
10. WAP in JavaScript to print the GCD of two number. (A)
11. Write a JavaScript to take 2-digit number and then separate these 2 digits, then multiply frst digit by itself for second digit times. (B)
(For example, 23 should be separated as 2 and 3. 2 should multiply with itself 3 times).
12. Write an HTML page with JavaScript that takes a number from popup box in the range 0-999 and display it in words. If the number is out of range, it should show “out of range” and if it is not a number ,it should show “not a number” message in the result box. (A)
13. Write an HTML fle with JavaScript that fnds position of frst occurrence of vowel “a”, last occurrence of vowel “a” in a given word and returns the string between them. For example, ajanta- then script would return frst occurrence of “a”-that is position 1 and last occurrence-6 and string between them is “jant”. (B)

Lab 02: Revision - ES6

1. WAP to demonstrate callbacks in JavaScript. (A)
2. Demonstrate the difference between let and var. (A)
3. Demonstrate the default parameter while using a function. (A)
4. Demonstrate the spread operator. (A)
5. Demonstrate the ‘for of’ loop. (A)
6. Demonstrate the Array and Object Destructuring. (B)
7. Demonstrate the Arrow functions. (B)
8. Demonstrate how to create a class in Java Script. (B)
9. Create a Snake game using Java script. (C)
10. Write JavaScript that handles following mouse event. (B)
• If mouse left button pressed on browser, it displayed message “Left Clicked”.
• If mouse right button pressed on browser, it displayed message “Right Clicked”.
11. Write a JavaScript having a list of checkbox and by clicking on checkbox, it should show list of selected value in comma separated format. (e.g. list of roll number as checkbox value, and display selected roll number in comma separated format) (C)

Lab 03: Practice NPM CLI commands

1. Create a new folder npm-lab-1 and initialize a Node project using npm. (A)
2. Compare the difference between npm init and npm init -y (A)
3. Install the package named “cors” using npm. (A)

4. Install the package named “nodemon” globally using npm. (A)
5. Install the package named “express” with version “4.21.1” locally (A)
6. Update the previously installed old version of express (B)
7. Uninstall all the packages that are locally installed in this lab. (A)
8) Uninstall globally installed “nodemon” package. (A)
9) practice installing multiple packages using package.json (A)
10) write a detailed note on package.json (B)
11) List all local packages. (A)

Lab 04: Practice path and os core module

1) Write a Node.js program that prints the directory name, file name, file extension, and full path of the code file. (A)
2) Create a program that joins "users", "arjun", "documents", "project" into a single path using path.join(). (A)
3) Write a program to fix the path="//folder//subfolder////file.txt" using path.normalize() and display the clean normalised path. (A)
4) Write a program that checks whether the given path is absolute or relative paths. (A)
5) Write a Node.js program that uses path.resolve() to convert "folder", "subfolder", "app.js" into an absolute path. Print the final resolved path. (A)
6) Write a program that prints: (A)
• Operating system name
• Release version
• Platform
• Architecture
7) Write a program that prints the total memory and free memory in GB (B)
8) Write a Node.js program that prints details about the currently logged-in user in operating system. (B)
9) Write a program that prints how long the system has been running (uptime) in seconds and in hours. (B)
10) Write a Node.js program that prints: (B)
• Number of CPU cores
• Model name of each core
• Network interface details

Lab 05: Practice child_process, url, util and event core module

1) Write a Node.js program using child_process.exec() to run the shell command to check the current version of node. (A)
2) Write a program that uses child_process.spawn() to run the command to print files and folders of current directory. (B)
3) Write a program to print current working directory using nodes. (B)
4) Write a program that parses the given URL, Print protocol, hostname, pathname, and query parameters separately. (A)
5) Create a new URL object with base, then append pathname and query, also print the final URL. (B)

6) Create an EventEmitter instance, Register an event "greet" and print a message when triggered. Fire that event manually using .emit().(B)

Lab 06: practice fs core module.

1) Write a Node.js program that uses fs.readFile() to read a file named data.txt asynchronously and print its content on the console. (A)
2) Use fs.readFileSync() to read info.txt and print the content, Compare execution with the asynchronous version. (A)
3) Create a program that writes the text into a file named output.txt. (A)
4) Create a program that appends the text into a file named output.txt. (A)
5) Write a program to delete a file named temp.txt using fs.unlink() and display success or error. (B)
6) Write a program that creates a folder named my-data using fs.mkdir(). If the folder already exists, show an appropriate message. (B)
7) Write a program to list all files in a folder called documents/ using fs.readdir() and print the file names one by one. (B)
8) Write a program that copies a file named source.txt to a new file named backup.txt using fs.copyFile(). (C)
9) Write a program that checks if the file config.json exists in the current directory using fs.existsSync() and prints the result. (B)
10) Write a program using fs.watch() to monitor changes in watchme.txt. Whenever file content changes, print: “File Changed” (C)

Lab 07: Create HTTP server using node.

1. Create a hello world webapp using “http” core module in NodeJS. (A)
2. Create a webapp with 5 pages like about, contact etc.. using “http” core module in NodeJS. (B)
3. Create a webapp in NodeJS which reads files like about.txt, contact.txt and display it using http core module (C)

Lab 08: Create express server with basic routing.

1. Create a hello world webapp using ExpressJS. (A)
2. Create a webapp with 5 pages like about, contact etc.. using ExpressJS. (B)
3. Create a webapp in NodeJS which reads files like about.txt, contact.txt and display it using express (C)

Lab 09: Create a MongoDB collections

1. Install Mongoose library using NPM. (A)
2. Demonstrate the use mongoose functions. (A)
3. Create a Database using MongoDBCompass for faculty. (A)
4. Create a Database using MongoDBCompass for student. (B)
5. Create a Database using MongoDBCompass for product. (C)

Lab 10: Setup MongoDB and middleware in ExpressJS

1. Demonstrate the use of middleware in Express. (A)
2. Demonstrate the use of static middleware in Express. (A)
3. Install MongoDB and MongoDBCompass (A)
4. Setup documents in MongoDB. (A)

Lab 11: Create CRUD API using Express, MongoDB and NodeJS

1. Create a restful CRUD API using NodeJS, Express and MongoDB for faculty. (A)
2. Create a restful CRUD API using NodeJS, Express and MongoDB for student. (B)
3. Create a restful CRUD API using NodeJS, Express and MongoDB for product. (C)

Lab 12: Create a mini project for library management system

Create and consume Restfull API using MongoDB, Express, ReactJS and NodeJS (MERN stack) for library management system.

Lab 13: Controllers in NestJS

1. Create a student controller with helloWorld method which returns “hello world” (A)
2. Create a student controller with findAll, findOne, insert, update and delete method and return appropriate string, make sure you use relevant HTTP methods. (A)
3. Create 3 different controllers with all above specified methods. (C)

Lab 14: Routing

1. Demonstrate @All decorator. (A)
2. Demonstrate @HttpCode decorator. (A)
3. Demonstrate @Redirect decorator. (A)
4. Demonstrate @Header decorator. (A)
5. Demonstrate Route Wildcards. (A)
6. Demonstrate Route Parameters,

· Write a code to get id from the parameters and print it. (A)
· Write a code to get start and end from parameters and return all the prime numbers between that range. (B)
· Write a code to get pageNo as parameter and return start and end record number, consider 5 records per page. (C)

Lab 15: Providers

1. Create a student service with findAll, findOne, insert, update and delete method to return “Hello world”. (A)
2. Write a NestJS application to combine student controller and service. (A)

3. Demonstrate module in NestJS. (A)

Lab 16: Resoruces

1. Createae a student resource. (A)
2. Implement static array CRUD API for students (A)
3. Implement static array CRUD API for 3 different array. (C)

Lab 17: Creating NestJS API

1. Create complete Rest API for students table stored in MySQL with following fields, (A)
- StudentID
- StudentName
- StudentAge
- StudentGender
- StudentRollNo
- StudentSemester.
- Etc...
2. Create a complete authentication API with login, register features (B)
3. Implement forgot password feature API. (C)

Lab 18: Perform Testing in NestJS

Perform Automated Testing in NestJS using Jest.

Lab 19: NextJS application setup and project sturecture

1. Setup a NextJS project using ""npx create-next-app@latest"" command. (A)
2. Explore Default Project Structure for NextJS created with create-next-app script. (A)
3. Update first page.tsx file to display ""Hello World from Next Application"". (A)
4. Write a detailed note on project structure of a NextJS application. (B)

Lab 20: Basic Routing and Navigation in NextJS application

1. Create a NextJS application with home, contact and about page. (A)
2. Create a NextJS application with basic Layout. (A)
3. Create a NextJS application with layout using bootstrap. (B)
4. Create a NextJS application with navigation between different pages. (B)

Lab 21: Apply CSS on NextJS application

1. Apply CSS on the static website created in the previous lab. (A)
2. Implement HTML/CSS template into NextJS application. (B)
3. Implement minimum 3 different template into NextJS application. (C)

Lab 22: Route group and dynamic routes in NextJS

1. Demonstrate the Route Group feature of NextJS.
· Create auth and admin route group. (A)
· Create auth, admin, client route group and apply different layout on each route group. (C)
2. Demonstrate the Dynamic Route feature of NextJS.
· Create a route with id as a parameter and print the id. (A)
· Create a route with start and end as a parameter and print the prime number between. (A)
· Create a route with pageNo as a parameter and print the start and end index of the record, assume we are getting 10 record per page. (C)

Lab 23: intercepting routes, route handler

1. Demonstrate the use of intercepting routes. (A)
2. Demonstrate the use of route handler. (A)
3. Create Rest API to perform CRUD operation on array using route handler. (B)

Lab 24: middleware, configuring NextJS application

1. Create a middleware in NextJS to add pageNo to be zero if it does not have any parameter in the request. (A)
2. Create a middleware to check for the token in the request, if it does not have token redirect them to login page. (B)
3. Create a middleware to get new token if the received token is expired. (C)
4. Explore NextJS configurations. (A)

Lab 25: Fetching Data from mockapi

1. Write a NextJS application to consume getALL api from mockapi and display the data on the page. (A)
2. Write a NextJS application to consume getByID api from mockapi and display the data on the page. (A)
3. Write a NextJS application to consume getALL and getByID api from mockapi and display on different pages, also provide navigation between them. (B)
4. Write a NextJS application to consume getALL and getByID operation for minimum 3 different APIs. (B)
5. Write a NextJS application to consume getALL and getByID operation for minimum 10 different APIs. (C)

Lab 26: Connect with database from NextJS

1. Write an application in NextJS to fetch the data from mysql. (A)
2. Write an application in NextJS to fetch the data from postgres. (A)
3. Write an application in NextJS to fetch the data from MongoDB. (B)

4. Write an application to perform getAll, getByID and search operation on a table stored in MySQL database. (B)
5. Write an application to perform getAll, getByID, search and getTasksByUserID operation on a database stored in MySQL with following tables (C)
- User table with UserID, UserName, Password
- Task table with TaskID, TaskTitle, TaskDescription, IsCompleted, UserID

Lab 27: Connect with database using Prisma ORM

1. Download and initialize Prisma ORM. (A)
2. Explore Prisma methods. (A)
3. Using Prisma ORM write an application to perform getAll, getByID, search and getTasksByUserID operation on a database stored in MySQL with following tables (A)
- User table with UserID, UserName, Password
- Task table with TaskID, TaskTitle, TaskDescription, IsCompleted, UserID

Lab 28: Server Actions

1. Write an application in NextJS to demonstrate Server action. (A)
2. Create server action to print the form data. (A)
3. Create server action on a separate file and use it in your component. (B)
4. Create a server action and validate the form data. (C)

Lab 29: Client Component and delete operation

1. Write an application in NextJS to create a client component (A)
2. Write an application in NextJS to create a basic calculator. (A)
3. Create a basic snake game using NextJS's client component (B)
4. Write an application to perform delete operation using Prisma ORM on a table from previous lab. (A)

Lab 30: Insert and Update the records using Prisma ORM with Automated Testing.

1. Create a registration form to insert data into database using prisma ORM. (A)
2. Create a form to add task to database table using Prisma ORM with following fields TaskID, TaskTitle, TaskDescription, IsCompleted, UserID. (B)
3. Create a form to change password into database using Prisma ORM. (A)
4. Create a form to edit Task stored in database. (B)
5. Write a test case to perform unit testing on NextJS application using Vitest. (A)
6. Write a test case to perform testing on NextJS application using Jest. (A)
7. Write a basic test cases to perform e2e testing on NextJS application using Playwrite. (A)
8. Write a test case to fill a form to add user and generate the report of the same using Playwrite. (B)