Section A: Short Answer Questions (2 × 5 = 10 Marks)

1. Explain the role of backend systems in modern web applications. (Unit I)

   The role of backend systems in modern web application is to securely manage data by storing, manipulating and retrieving data using databases. It applies business logic and generates outputs for the frontend. It also ensures security through role-base authorization, encrypted data storage and secure API practices. Additionally, backend also helps in performance optimization through load balancing, caching, asynchronous processing and optimizing database queries using efficient data structures.

2. Define the three layers in a multi-tiered architecture. (Unit I)

   Three layers in a multi-tiered architecture are:

1. Presentation Layer: It is the user interface which display information and captures user inputs.
2. Business Logic Layer: It processes the data, applies business logic rules and make decisions before passing the data to or from database.
3. Data Layer: It manages the storage and retrieval of data using databases, handles quires and ensure data is accessible.

3. What are the advantages of Node.js being event-driven and non-blocking? (Unit II)

The advantages of Node.js being event-driven and non-blocking is that it allows multiple requests at the same time without waiting for the previous task to complete which makes it efficient, scalable and fast for the real-time applications like live updates. It uses system resources more effectively and can improve overall performance.

4. What is an MVC framework? List its three core components. (Unit III)

The MVC framework is a software framework used for building web application by separating a web application into three parts namely Model, View and Controller. The Models manages data, database interactions and business logic. The View handles the user interface. The Controller act as a intermediary between model and view by processing user inputs, updating data in the Model and refreshing the View to show the new information.

5. What is the difference between require() and import in Node.js? (Unit II)

Require() in Node.js	**Import in Node.js**
Is a synchronous process and load one module at a time.	Is an asynchronous process and can load module in the background.

Section B: Medium Answer Questions (5 × 6 = 30 Marks)

1. Compare frontend and backend development with suitable examples. (Unit I)

Frontend(Client-Side)	Backend(Server-side)
Is the part of web application which interacts with the user directly.	Is the hidden part of the web application that handles data and business logic.
Uses technologies like CSS, HTML, JavaScript and framework like Vue.js, React and Angular	Uses technologies like Django, Flask, Node.js and databases like MySQL, MongoDB and PostgreSQL.
It involves design, button, text and layout shown on the screen.	It involves handing requests from frontend, processing data and managing database interactions.

Example: Virtual Learning Environment

The fronted of the VLE is what the lecturers and students interact directly. When students log in, they can see a dashboard that shows their course, grades and assignment. They can click on course to view course materials like reading materials, lectures and multimedia content. Students can participate in discussion forum, submit their assignment and track their academic progress. The frontend collects user inputs and display the data in user-friendly way.

The backend of the VLE is responsible for managing the data and processing request from the frontend. When students submit an assignment, the backend store the data and notifies the instructor. It also makes sure that the grades are properly stored and student can view their result once they are graded. The backend handle course management and also manages user authentication to ensure that only the registered student can access the course.

2. Describe the Node.js ecosystem. Include key tools and how packages are managed using npm. (Unit II)

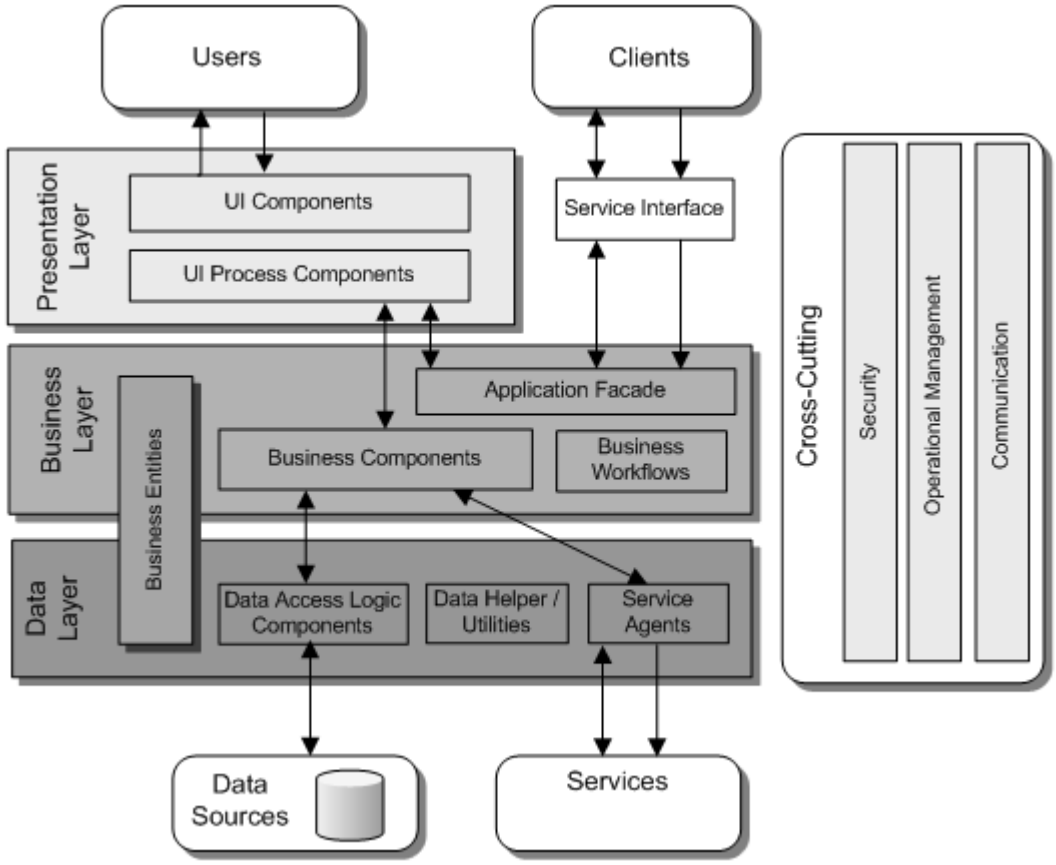
The Node.js ecosystem is an environment for building backend application using JavaScript. It consists of rich set of libraries, frameworks and tools that allows developers to build scalable and efficient web application. One of the key features of Node.js is its non-blocking, event-driven architecture which helps in handling multiple requests. Some key tools of Node.js are:

1. Npm: is an essential tool to manage packages that the developer use in their applications. With the help of npm, developers can easily install, update and manage dependencies.
2. Express.js: is a flexible web application framework for Node.js. It simplifies HTTP request, routing and middleware for better application performance.
3. Nodemon: is a tool to automatically restart Node.js application whenever code changes are found.

The packages are managed using npm by:

1. Installing Packages: npm install package-name
2. Uninstalling Packages: npm uninstall package-name
3. Updating Packages: npm update
4. Viewing Installed Packages: npm install
5. Saving Dependencies: npm install package-name - -save-dev

8. Illustrate with a diagram how the presentation layer, business logic layer, and data layer interact in a web application. (Unit I)



In a web application, the Presentation Layer Is the frontend part where the user interacts. It is build using HTML, CSS and JavaScript framework like Angular and React. When a user perform action, the request is sent to the business logic layer. Business logic layer is powered by backend technologies like Express.js and Node.js which are responsible for processing the request. It applies business rules, manage authentication, handles validations and decides what need to happen next. If the request includes storing, retrieving and updating data, the Business logic layer communicates with the Data Layer which consist of databases. After the data is fetched by Data Layer, it moves back through the business logic layer and is then send to the presentation layer to display the updated information to the users. This structure interaction ensures the application is organized, scalable and easier to maintain.

9. Explain the purpose of the MVC pattern and how it helps in organizing web application code. (Unit III)

The MVC pattern is a way to organize code to make the building of web application cleaner, easy to manage and scalable. It is separated into three parts:

1. Model: It handles data and business logic. It talks to database, store information and manages rules.
2. View: It controls how the information is presented to the user. It includes the frontend part where we use HTML, CSS and template for user to interact.
3. Controller: It acts as intermediary between the model and View where it takes user request, processes them and return to the View where user interact.

How MVS helps in organizing code:

1. Easier Maintenance: In MVC all the code are separate which makes it easy to fix bug or add new feature without breaking the other part.
2. Separation of Concerns: Each part has a clear responsibility where the model doesn't mix with the view, and controller only act as an intermediary between model and view.
3. Scalability: As the application grows, it will be easier to manage as each section is independent and organized.
4. Flexible Development: Changing code in one layer doesn't force big changes in the other layer which makes it easy to update the application without starting from scratch.
5. Improve Security: By separating the View layer from the model layer, its easier to implement security checks and validations, reducing data leaks.

10. Outline the steps to set up Node.js on a POSIX-like system. Include how to verify a successful installation. (Unit II)

To set up Node.js on a POSIX-like system, the first step is to update the system's package list to ensure that all existing packages are up to date. Updating system's packages can be done by running the code "sudo apt update" in the bash or terminal. Updating the package list can help developers avoid potential issues during installation.

After updating the system, then the next step is to install Node.js along with npm. This can be achieved by running the command "Sudo apt install nodejs" for Node.js and "sudo apt install npm" for npm. Node.js allows developers to run JavaScript code outside of a browser, while npm is important for managing JavaScript libraries and packages.

Then we have to verify whether the Node.js and npm are successfully installed or not. It can be done by checking their version by running the code "node -v" and "npm -v". If the version numbers are displayed without any error, it confirms that the installation was successful.

After that, we have to test Node.js to check its functionality by creating a simple program. The test is done to check and confirm that Node.js can run code properly on the system.

Section C: Long Answer Questions (2 × 10 = 20 Marks)

11. Discuss the key reasons for using Node.js in backend development. How does it compare to traditional backend technologies? (Unit II)

Some of the key reason for using Node.js in the backend development are:

1. Non-Blocking, Asynchronous I/O: Node.js is designed with non-blocking architecture which can handle multiple requests at the same time without waiting for the previous request to finish.
2. Single Language across frontend and backend: with the help of Node.js, developer can use JavaScript for both frontend and backend.
3. Event-driven Architecture: Node.js uses event loop to manage operations which makes it fast and lightweight. Node.js can handle many connections on a single thread by managing events efficiently instead of creating new threads for each request.
4. Fast performance: Node.js uses Google's V8 JavaScript Engine, which compiles JavaScript into machine code resulting into extremely fast execution and making backend application more scalable.
5. Huge Ecosystem with npm: Node.js has a many package manager(npm) that can be easily installed, saving developers time and effort. With the help of npm, we can find ready mad e solution for authentication, database handling, file upload and APIs.
6. Real time Application Support: Node.js is perfect for building real-tine application such as chat application, live streaming and online gaming.

Difference between Node.js and Traditional Backend

Feature	**Node.js**	**Traditional Backend**
Language	Uses JavaScript for both frontend and backend	Need different language for frontend and backend.
Speed	Very Fast	Slower compare to Node.js
Learning Curve	Easier for JavaScript developers	Separate learning needed
Development Time	Faster	Slower
Scalability	Excellent	Good but more complex
Ecosystem	Has massive npm library	Large but different for each language
Real-time Support	Built-in	Need extra libraries and tools
Concurrency	Non-blocking I/O	Blocking I/O

12. Design a basic blueprint for a blogging platform using the MVC pattern. Explain how each part (Model, View, Controller) will handle different tasks. (Unit III)

Model: represent the database structure and operation related to data. In a blogging platform, the model will store blog post, store user details, manage comments and handles like, categories and tag.

Important Model Files:

File	Purpose
postModel.js	Define blog post schema.
userModel.js	Define user schema.
commentModel.js	Define comment schema.
categoryModel.js	Define categories for organizing blog posts.

View: The view is the frontend part where the user interacts like web page, forms and blog list. In the blog platform, the view will handle displaying the homepage with all blog post, showing individual blog post with comment, providing forms to create or update a blog post, showing signup and login form, and showing success or error message.

Important Views files:

File	Purpose
Home.ejs	Will display all blog post on homepage
Post.ejs	Display a blog post with comments. Can create and edit blog post.
Signup.ejs	Signup form for new user.
Login.ejs	Login form for existing users.
Profile.ejs	User profile page showing their posts.
Layout.ejs	Common layout for all pages.

Controller: The controller handles the logic by connecting Model and the View. It's role is to take user inputs form the View, process them and use the appropriate Model methods, and then decide which View should be rendered. The Controller's responsibilities include validating the data, creating or updating blog post, handle user authentication, fetch posts to display on the homepage and manage actions like deleting or adding comment.

Important Controller Files:

File	Purpose
postController.js	Manage blog post operations (create, edit, delete, fetch posts)
userController.js	Manage user actions like signup, login and logout
commentController.js	Manage comment operations (add, delete, fetch comments)
categoryController.js	Manage blog categories like add, delete and list.

Routes: Routes are responsible for mapping URL request to the appropriate Controller. Routes keep the application organized and ensure that each URL endpoints triggers the correct functionality. Routes responsibilities include defining URL path and linking them to function inside controllers.

Important Routes Files:

File	Purpose
postRoutes.js	/posts, /posts:id, /create-post
userRoutes.js	/signup, /login, /logout, /profile
commentRoutes.js	/posts:id/comments, /comments/delete/:id
categoryRoutes.js	/categories, /categories/:id

Public: The Public folder serves all the static files like CSS, Client-side JavaScript and images. These are required to make the application visually appealing and interactive. It provides styling directly on the

fronted without any server-side processing. Its responsibilities include providing stylesheets for page design, JavaScript files for frontend functionalities and images for user profile or blog posts.

Important public structure:

Folder	Purpose
/css/style.css	Website layout and design
/Js/main.js	Client-side interactivity
/images/	Store profile pictures and blog images.

0ther necessary components:

Folder/file	Purpose
Config/database.js	Set up database connection
app.js	Main entry file, initialize server, setup middleware and mount routes.
Package.json	Project metadata and dependency management.