Sub:	**Web Development using Full Stack Open - Solution**

Note : Answer FIVE FULL Questions, choosing ONE full question from each Module

PART I

1 Explain the core principles of Redux in detail.

OR

2 Describe npm and its package management features.

PART II

3 Write and explain the steps for building a basic server using Express.

OR

4 Explain the architecture of Node.js in detail.

PART III

5. What is MongoDB? Explain NoSQL concepts and compare MongoDB with relational databases

OR

6 Explain different types of middleware functions such as logging, authentication, and error handling.

PART IV

7 Describe RESTful API principles in detail and explain the best practices for designing REST APIs..

OR

8 Explain the architecture and components of a full-stack application.

PART V

9 Explain the concept and workflow of Continuous Integration and Continuous Deployment (CI/CD) pipelines.

OR

10 Compare Jenkins and GitHub Actions as CI/CD tools.

Solutions

1. Explain the core principles of Redux in detail.
Redux is a predictable state management library used mainly with React applications to manage global state.
The three core principles of Redux are:

1. Single Source of Truth:
The entire application state is stored in a single JavaScript object called the store. This makes debugging easier and ensures consistency across the application.

2. State is Read-Only:
The state cannot be changed directly. The only way to change the state is by dispatching actions, which are plain JavaScript objects describing what happened.

3. Changes are Made with Pure Functions:
Reducers are pure functions that take the current state and an action as arguments and return a new state. They do not modify the existing state.

Redux ensures predictable behavior, easier debugging, and scalability for large applications.

Redux Data Flow:

- UI dispatches an action
- Reducer processes the action
- Store updates the state
- UI re-renders with updated state

Advantages:

- Predictable behavior
- Easier debugging with Redux DevTools
- Scalable for large applications

2. Describe npm and its package management features.

npm (Node Package Manager) is the default package manager for Node.js used to install, manage, and share JavaScript libraries.
Key Features of npm:

1. Package Installation
   - Install libraries using npm install package-name.
2. Dependency Management
   - Dependencies are listed in package.json.
   - Ensures consistent environments across systems.
3. Version Control
   - Semantic versioning (major.minor.patch) avoids breaking changes.
4. Scripts Support
   - Allows defining custom scripts such as npm start, npm test.
5. Global and Local Packages
   - Local: project-specific
   - Global: system-wide tools like nodemon

Advantages:

- Large ecosystem
- Easy dependency handling
- Strong community support

.

3. Write and explain the steps for building a basic server using Express.
Express is a minimal Node.js framework used for building web servers.

Steps:

1. Initialize project using npm init
   npm init -y
2. Install Express using npm install express
   npm install express
3. Create server file and import Express
   ```
   const express = require('express');
   const app = express();

   app.get('/', (req, res) => {
     res.send('Hello Express');
   });
   app.listen(3000, () => {
   console.log('Server running on port 3000');
   });
   ```
4. Define routes using app.get()
   node app.js

5. Start server using app.listen()

- express() creates the app
- app.get() handles HTTP GET requests
- app.listen() starts the server

Express simplifies routing and request handling.

4. Explain the architecture of Node.js in detail.

Node.js is a runtime environment that executes JavaScript on the server side using a non-blocking architecture.
Key Components:

1. V8 Engine
   - Compiles JavaScript into machine code.
2. Event Loop
   - Handles asynchronous operations efficiently.
3. Callback Queue
   - Stores completed async operations.
4. Non-Blocking I/O
   - Prevents thread blocking during file or network operations.
5. Single Threaded Model
   - Uses one main thread but handles multiple requests efficiently.

Advantages:

- High performance
- Scalable
- Ideal for real-time applications

5. What is MongoDB? Explain NoSQL concepts and compare MongoDB with relational databases.

MongoDB is a NoSQL, document-oriented database that stores data in JSON-like documents.

NoSQL Concepts:

- Schema-less design
- Horizontal scalability
- High availability

Feature	**MongoDB**	**Relational DB**
Data Model	Document	Table
Schema	Flexible	Fixed
Scalability	Horizontal	Vertical
Joins	Limited	Strong

Use Cases:

- Big data
- Real-time applications
- Content management

6. **Explain different types of middleware functions.**

Middleware functions execute between request and response.

Types:

1. Logging middleware: Logs requests

```
app.use((req, res, next) => {
 console.log(req.method);
 next();
});
```

2. Authentication middleware: Verifies users

3. Error-handling middleware: Handles application errors

```
app.use((err, req, res, next) => {
 res.status(500).send('Error occurred');
});
```

Middleware improves modularity and maintainability.

Purpose:

- Code reusability
- Request preprocessing
- Centralized error handling

7. Describe RESTful API principles and best practices.
REST (Representational State Transfer) is an architectural style for building APIs.
Principles:

- Stateless communication
- Resource-based URLs
- HTTP methods (GET, POST, PUT, DELETE)

Best Practices:

- Use meaningful endpoints
- Proper status codes
- Version APIs (/api/v1)
- Secure with authentication

8. Explain the architecture and components of a full-stack application.
A full-stack application consists of both front-end and back-end components.
Components:

1. **Front-End**
- React / HTML / CSS
- Handles UI and user interaction
2. **Back-End**
- Node.js / Express
- Business logic and APIs
3. **Database**
- MongoDB / MySQL
- Persistent data storage
4. **API Layer**
- Connects front-end and back-end

9. Explain the concept and workflow of CI/CD pipelines.
CI/CD automates software development and deployment.
CI (Continuous Integration):

- Code integration
- Automated testing

CD (Continuous Deployment):

- Automated deployment
- Faster release cycles

Workflow:
Code → Build → Test → Deploy
Benefits:

- Reduced errors
- Faster delivery
- Improved quality

10. Compare Jenkins and GitHub Actions as CI/CD tools.

Feature	Jenkins	GitHub Actions
Setup	Complex	Simple
Integration	Any repo	GitHub only
UI	Plugin-based	YAML-based
Maintenance	Manual	Managed