V Semester Diploma Makeup Exam July/August 2024

Full Stack Development-20CS52I

Scheme of Valuation

SECTION I

1a) Definition of/About Digital transformation: Each point on digital transformation in healthcare industry carries 2 marks. 2x5marks=10marks

1b) Definition 2marks, Explanation -8 Marks. 2+8marks=10marks

2a) Explaining the process of design thinking 5 marks, Apply the process of design thinking that led to the evolution of electric vehicles (each carries 1mark)5marks. 5+5=10marks

2b) Listing Each difference for all three, carries 2 marks. 2 x 5=10marks

SECTION II

3a) Writing of any 5 user stories with Acceptance criteria=5x2=10 Marks

3b) Writing any 5 test cases=5x2=10 Marks

4a) Writing of any 5 user stories with Acceptance criteria=5x2=10 Marks

4b) Writing any 5 test cases=5x2=10 Marks

SECTION III

5a) Explaining States with example=5 Marks, Explaining Props with example=5 Marks

5b) Entity class -2 Marks, 4 controller carries 2 marks each – 4x2=8 Marks. 2+8=10marks

6a) any 5 Comparison – 5x2=10 Marks

6b) Implementation - 10 Marks

SECTION IV

7a) Each annotation explanation carries 2marks – 5x2=10 Marks

7b) Each MongoDb operation carries 2 marks – 2x5=10 Marks

8a) Acid properties mentioned – 2marks + explanation of acid properties 8marks – 10marks

8b) Spring Security Explanation -5 Marks, Code & explanation -5 Marks

SECTION V

9a) any 5 Comparison – 5x2=10 Marks

9b) Explanation of each point 5x2marks=10 marks

10a) Diagram=4 Marks, Listing and explaining all 6 components=6 Marks

10b) Bluegreen deployment with diagram 5 marks, canary deployment with example 5marks. (Diagram each carries 1 mark)

Model Answers

Section-I

1. a) Explain how digital transformation can bring revolution in healthcare industry. 10 Marks

Digital transformation can profoundly revolutionize the healthcare industry by fundamentally changing how care is delivered, managed, and experienced. Ways in which digital transformation can bring about this revolution**:**

1. Enhanced Patient Experience

- Telemedicine: Provides remote access to healthcare services, allowing patients to consult with doctors from their homes. This improves accessibility for people in remote areas and reduces waiting times.

- Patient Portals: Enable patients to access their medical records, schedule appointments, and communicate with their healthcare providers online, leading to greater convenience and engagement.

2. Improved Diagnostics and Treatment

- Artificial Intelligence (AI): AI-powered tools assist in interpreting medical images, predicting patient outcomes, and personalizing treatment plans. For example, AI algorithms can detect early signs of diseases such as cancer from imaging data with high accuracy.

- Precision Medicine: Uses data from genetic information, lifestyle factors, and other sources to tailor treatments specifically to individual patients, improving effectiveness and reducing adverse effects.

3. Operational Efficiency

- Electronic Health Records (EHRs): Digital records streamline the management of patient information, enhancing coordination among healthcare providers and reducing errors associated with manual record-keeping.

- Automation: Automates administrative tasks such as billing, scheduling, and inventory management, reducing costs and freeing up staff to focus more on patient care**.**

4. Data-Driven Decision Making

- Big Data Analytics: Analyzes large volumes of data to uncover trends, predict health outcomes, and support evidence-based decision-making. This can lead to better disease prevention strategies and more effective treatments.

- Predictive Analytics: Uses historical data and machine learning to anticipate health issues before they become critical, enabling earlier intervention and improved patient outcomes.

5. Enhanced Patient Engagement

- Wearable Technology: Devices like smartwatches and fitness trackers collect real-time health data, allowing for continuous monitoring of chronic conditions and providing valuable insights to both patients and healthcare providers.

- Mobile Health Apps: Help patients track their health metrics, manage medications, and access health education, empowering them to take an active role in their care.

6. Cost Reduction

- Remote Monitoring: Reduces the need for in-person visits, which can lower costs for both patients and healthcare systems. It also enables timely interventions that can prevent costly emergencies.

- Resource Optimization: Digital tools help in better resource allocation and reducing unnecessary tests and procedures, leading to more cost-effective care.

7. Better Access to Care

- Global Health Initiatives: Digital platforms facilitate collaboration and knowledge sharing across borders, improving the ability to tackle global health challenges and expand access to care in underserved areas.

- Mobile Clinics and Health Apps: Extend healthcare services to populations with limited access to traditional facilities**.**

8. Enhanced Security and Compliance

- Cybersecurity: Advances in digital security ensure that patient data is protected from breaches and unauthorized access, maintaining confidentiality and compliance with regulations.

9. Faster Innovation and Research

- Clinical Trials: Digital platforms accelerate the recruitment process and data collection for clinical trials, leading to faster development of new treatments and therapies.

- Collaboration Tools: Facilitate research collaboration and data sharing among scientists, clinicians, and institutions, speeding up the development of new medical knowledge and technologies.

Note: any related point may be considered.

1 b) Define Enterprise. Explain the steps involved in the organizing an enterprise. 10 marks.

An enterprise is a project, a willingness to take on a new project, an undertaking or business venture. An example of an enterprise is a new start-up business or someone taking initiative to start a business.

Organizing the Enterprise process –

Five main steps involved in the process of organizing an enterprise is

1. Determining Activities

2. Grouping of Activities

3. Assigning Duties

4. Delegating Authority

5. Coordinating Activities.

1. Determining Activities

❖ The first step in organizing is to identify and enumerate (to specify one after another) the activities required to achieve the objectives of the enterprise.

❖ The activities will depend upon the nature and size of the enterprise.

❖ For instance, a manufacturing concern will have production, marketing and other.

2. Grouping of Activities

❖ The various activities are then classified into appropriate departments and divisions on the basis of functions, products, territories, customers etc.

❖ Similar and related activities may be grouped together under one department or division.

❖ Grouping of activities helps to secure specialization. Each department may be further sub divided into sections and groups.

3. Assigning Duties

❖ The individual groups of activities are then allotted to different individuals on the basis of their ability and aptitude.

❖ The responsibility of every individual should be defined clearly to avoid duplication of work and overlapping of effort.

❖ Each person is given a specific job best suited to him and he is made responsible for its execution.

4. Delegating Authority

❖ Every individual is given the authority necessary to perform the assigned task effectively.

❖ An individual cannot perform his job without the necessary authority or power.

5. Coordinating Activities

❖ The activities and efforts of different individuals are then synchronized. Such co-ordination is necessary to ensure effective performance of specialized functions.

2a) Explain the process of design thinking. Apply the process of design thinking that led to the evolution of electric vehicles. 10 marks

Design thinking is a human-centered approach to innovation that emphasizes understanding users' needs, defining problems, generating creative solutions, prototyping, and testing. It's a flexible process that fosters collaboration and iteration to address complex challenges. This process involves:

1. Empathize: Gain a deep understanding of the users and their needs through observation and engagement.

2. Define: Clearly articulate the problem based on insights from the empathize phase.

3. Ideate: Brainstorm a wide range of ideas and solutions to address the defined problem.

4. Prototype: Create tangible representations of ideas to explore how they might work in practice.

5. Test: Evaluate prototypes with users to gather feedback and refine solutions.

6. Implement: Finalize and launch the solution, integrating feedback and ensuring it meets user needs effectively.

Applying Design Thinking to the Evolution of Electric Vehicles

1. Empathize

The evolution of electric vehicles (EVs) began with understanding the problems and needs of both consumers and the environment. Key insights included:

- Environmental Concerns: There was growing awareness about the negative impact of fossil fuel-powered vehicles on climate change and air quality.

- Consumer Frustrations: Drivers faced issues such as high fuel costs, maintenance requirements, and limited efficiency of traditional internal combustion engine (ICE) vehicles.

- Technological Challenges: Engineers and designers recognized the need for advancements in battery technology and electric drivetrains to make EVs viable.

2. Define

The core problem was identified as the need for a more sustainable and efficient mode of transportation. The focus was on:

- Reducing Emissions: Addressing the environmental impact of conventional vehicles.

- Improving Efficiency: Enhancing energy efficiency and reducing reliance on fossil fuels.

- Meeting Consumer Needs: Creating a vehicle that is affordable, reliable, and offers a comparable or superior driving experience to ICE vehicles.

3. Ideate

During this phase, a variety of innovative ideas were explored:

- Battery Technology: Development of high-capacity, fast-charging batteries to extend range and reduce charging time.

- Vehicle Design: Designing vehicles that maximize aerodynamics and efficiency.

- Charging Infrastructure: Creating convenient and accessible charging solutions, including home chargers and public charging stations.

- Business Models: Exploring different ownership and leasing models to make EVs more accessible to a broader audience.

4. Prototype

Prototypes were developed to test and refine ideas:

- Early EV Models: Companies like Tesla and Nissan created prototype electric cars to test battery performance, vehicle range, and user acceptance.

- Charging Solutions: Prototypes of charging stations and home charging units were developed to address infrastructure needs.

- User Interfaces: Prototypes included user-friendly interfaces for managing charging, monitoring battery health, and providing navigation.

5. Test

Prototypes were tested extensively to gather feedback and make improvements:

- User Feedback: Early adopters provided valuable insights on vehicle performance, range, and charging experience.

- Performance Testing: Vehicles were subjected to rigorous testing to assess battery life, reliability, and overall driving experience.

- Infrastructure Testing: Charging solutions were tested for convenience, speed, and accessibility.

6. Implement

Based on feedback from testing, the final solutions were refined and implemented:

- Mass Production: Electric vehicles moved from prototypes to production models, incorporating improvements based on testing.

- Infrastructure Expansion: Charging networks were expanded to support the growing number of EVs on the road.

- Market Adoption: Continuous improvements in battery technology, vehicle design, and charging infrastructure contributed to increased adoption and market penetration.

2 b) Differentiate between 3 different cloud service models. 10 Marks

Basis of	IAAS	PAAS	SAAS
Stands for	Infrastructure as a service.	Platform as a service.	Software as a service
Uses	IAAS is used by network architects.	PAAS is used by developers.	SAAS is used by the end user.
Access	IAAS gives access to the resources like virtual machines and virtual storage.	PAAS gives access to run time environment to deployment and development tools for application.	SAAS gives access to the end user.
Model	It is a service model that provides virtualized computing resources over the internet.	It is a cloud computing model that delivers tools that are used for the development of applications.	It is a service model in cloud computing that hosts software to make it available to clients.
Technical understanding.	It requires technical knowledge.	Some knowledge is required for the basic setup.	No requirement about technicalities; company handles everything.
Popularity	It is popular among developers and researchers.	It is popular among developers who focus on the development of apps and scripts.	It is popular among consumers & company such as file sharing, email, and networking.
Usage	Used by the skilled developer to develop unique applications.	Used by mid-level developers to build applications.	Used among the users of entertainment.
Cloud services.	Amazon Web Services.	Facebook, and Google search engine.	MS-Office web, Facebook and Google Apps.
Enterprise services.	AWS virtual private cloud.	Microsoft Azure.	IBM cloud analysis.
User Controls	Operating System, Runtime, Middleware, and Application data	Data of the application	Nothing

Section-II

3a. KSRTC Awatar is an online bus ticket booking mobile application which helps passengers to book a bus ticket for travelling in KSRTC buses across Karnataka state. Users can Install app from the app store and book bus tickets. Users can create online account by signing up through app. Registered users can login and search for bus routes by entering source and destination places along with date of journey. Users can get bus details, seats availability and book/cancel tickets. Once ticket is booked by paying requisite amount through online payment gateway, user will be notified by SMS & email about booking details. Identify and write the user stories for this application. -10 marks

User story is an informal, general explanation of a software feature written from the perspective of the end user or customer.

For KSRTC Awatar application, the user stories are as follows-

1. User Story: Registration/Sign-up:

As a new user I want to sign up for the application through a sign-up form, so that I can access the online bus ticket booking app.

Acceptance Criteria:

i. While signing up, enter Username, Mobile Number, Email, Password & Confirm Password, Security question and Address.
ii. If user sign up with an incorrect detail, user receives an error message for incorrect information.
iii. If user tries to sign up with an existing email address, existing mobile number, user receives error messages saying "email exists", “mobile registered already”.
iv. If sign up is successful, a confirmation email is sent to user for mobile & email verification. After successful verification user can login into app with credentials

2. **User Story: Login**

As a registered/authorized user, I want to login into KSRTC bus booking application so that I can access variety of services provided by the app.

Acceptance Criteria:

i. Username, password and captcha are required for user login.
ii. If we are trying to login with incorrect username or password, then error message will be displayed as "invalid credentials”.
iii. After successful log in, home page/dashboard is displayed.

3. User Story: Search

As an authorized user, I want to search the buses from one place(source) to other place(destination) for a particular date so that I can book seat/s in a bus.

Acceptance Criteria:

i. User has to enter valid source, valid destination and valid date.
ii. A valid search displays list of buses available with seats & price information from a source place to destination place for a particular date.
iii. User can sort, filter and modify the search results.

4. **User Story: Book Bus Ticket**

As an authorized user, he/she wants to book a bus ticket so that he/she can travel from source place to destination place on specified date.

Acceptance Criteria:

- User has to choose among the available buses with seat for booking.

- User details like name, email, phone number, address, seat selection, pickup point, drop point etc need to be entered
- Valid payment mode is to be selected for making necessary payment
- After successful payment, user should get the booking details to registered mobile number and E-mail id

5. User Stories: Logout

As a logged in user, he/she wants to log out of KSRTC bus booking app so that I can prevent unauthorized access to user profile.

Acceptance Criteria:

i. When user logs out of his account by clicking log out button, logged out message should appear and app has to redirected to the log-in page.
ii. If user session expires due to internet failure or system crash, then user has to be logged out the application.

3 b) Write test cases for the above application. 10 Marks

Test Cases for the Registration Page:

i. Verify that the registration page is accessible from the website's homepage and loads correctly for desktop and mobile versions.
ii. Check that the system validates the user's information such as email address, phone number, security questions, address and password complexity.
iii. Ensure that the system does not allow duplicate email addresses or phone number
iv. Verify that the user receives an email or SMS confirmation after registering.

Test Cases for the Login Page:

i. Verify that the login page/app loads correctly and is accessible from the website's homepage.
ii. Check that the login credentials are case sensitive and the appropriate message is displayed if the user enters incorrect information.
iii. Verify that the "Forgot Password" option works as intended, allowing users to reset their password in case they forget it.
iv. Ensure that the system limits the number of unsuccessful logins attempt to prevent brute-force attacks.

Test Cases for the Buses Search:

i. Ensure that the Bus search page displays list of buses corresponding to given source, destination & date.
ii. Verify whether user is able to apply filter, sort buses and modify existing search.
iii. Verify whether user is able to select the required bus, check for availability of seats for a particular journey.

Test Cases for the Bus Ticket Booking:

i. Verify that the system displays the total cost of the seat/s as per booking details, including any taxes, discount and fees.
ii. Verify whether user is able to cancel the bus ticket or not.
iii. Verify whether booking confirmation is received by user or not.

Test Cases for the Payment Gateway:

i. Verify that the payment gateway is secure & encrypts user information to prevent fraud.
ii. Ensure that the system accepts multiple payment options, such as credit/debit cards, GPay, PhonePe and mobile wallets
iii. Ensure that the payment gateway sends a confirmation email or SMS to the user after the successful transaction

4a. Zomato is an online food ordering application that helps its users to buy variety of food items. This application allows users to log in for ordering food. Users can search for their favourite food based on rating or price. Uses can select the food items and add to cart. Once the food selection is finalized, user can pay online through online payment methods and order/cancel their food. Users will be notified about the order details & delivery of food. Identify and write the user stories for this application. 10marks

User story is an informal, general explanation of a software feature written from the perspective of the end user or customer.

For Zomato application, the user stories are as follows-

1. <u>User Story: Registration/Sign-up:</u>

As a new user I want to sign up for the application through a sign-up form, so that I can access the Zomato app.

Acceptance Criteria:

- While signing up, enter Username, Email, Password & Confirm Password, Security question and Address.
- If user sign up with an incorrect detail, user receives an error message for incorrect information.
- If user tries to sign up with an existing email address, existing mobile number, user receives error messages saying "email exists", “mobile registered already”.
- If sign up is successful, a confirmation email is sent to user for mobile & email verification. After successful verification user can login into app with credentials

2. <u>User Story: Login</u>

As a registered/authorized user, he/she wants to login into zomato application so that user can search and order food provided by various restaurants listed in the app.

Acceptance Criteria:

- Username, password and captcha are required for user login.

- If we are trying to login with incorrect username or password, then error message will be displayed as "invalid credentials".
- After successful log in, home page is displayed.

3. User Story: Search

As an authorized user, he/she wants to search the variety of food so that user can order the food of his/her choice.

Acceptance Criteria:

- User has to enter valid food name.
- A valid search displays list of restaurants with food available
- User can sort, filter and modify the search results based on the rating or price.

4. **User Story: Food Ordering**

As an authorized user, he/she wants to order a food of his/her choice from the list of available foods

Acceptance Criteria:

- User has to choose among the available food/s for booking.
- User details like name, email, phone number, address, etc need to be entered
- Valid payment mode is to be selected for making necessary payment
- After successful payment, user should get the order details to registered mobile number and E-mail id

5. User Stories: Logout

As a logged in user, he/she wants to log out of zomato app which prevents unauthorized access to user profile.

Acceptance Criteria:

- When user logs out of his account by clicking log out button, logged out message should appear and app has to redirected to the log-in page.
- If user session expires due to internet failure or system crash, then user has to be logged out the application.

4 b) Write the test cases for the above application. 10 Marks

Test Cases for the Registration Page:

i. Verify that the registration page is accessible from the website's homepage and loads correctly for desktop and mobile versions.
ii. Check that the system validates the user's information such as email address, phone number, security questions, address and password complexity.
iii. Ensure that the system does not allow duplicate email addresses or phone number
iv. Verify that the user receives an email or SMS confirmation after registering.

Test Cases for the Login Page:

i. Verify that the login page loads correctly and is accessible from the website's homepage.
ii. Check that the login credentials are case sensitive and the appropriate message is displayed if the user enters incorrect information.
iii. Verify that the "Forgot Password" option works as intended, allowing users to reset their password in case they forget it.
iv. Ensure that the system limits the number of unsuccessful logins attempt to prevent brute-force attacks.

Test Cases for the Food Search:

i. Ensure that the food search page displays list of restaurants with food.
ii. Verify whether user is able to apply filter, sort food and modify existing search.
iii. Verify whether user is able to select the required food, check for its availability and quantity.

Test Cases for the Food Ordering:

i. Verify that the system displays the total cost of the food as per ordering details, including any taxes, discount and fees.
ii. Verify whether user is able to cancel the food or not.
iii. Verify whether order confirmation is received by user or not.

Test Cases for the Payment Gateway:

i. Verify that the payment gateway is secure & encrypts user information to prevent fraud.
ii. Ensure that the system accepts multiple payment options, such as credit/debit cards, GPay, PhonePe and mobile wallets
iii. Ensure that the payment gateway sends a confirmation email or SMS to the user after the successful transaction

Section-III

5 a) Explain State and Props withg an example in ReactJS **10 marks**

• State in React JS

The state is a built-in React object that is used to contain data or information about the component. A component's state can change over time; whenever it changes, the component re-renders. The change in state can happen as a response to user action or system-generated events. The changes determine the behavior of the component and how it will render. The state object is initialized in the constructor. The state object can store multiple properties.

```
Class Car extends React Component {

Constructor(props) {

super(props);

this.state = {brand: "Ford"};
```

```
 }
render() {
return (<div> <h1>My Car</h1> </div>) ;
}}}
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<Car />);
```

• Prop in ReactJS

Props is short for properties and they are used to pass data between React components. React's data flow between components is uni-directional (from parent to child only). React Props are like function arguments in JavaScript and attributes in HTML. Props are immutable so we cannot modify the props from inside the component. Props are read-only components. It is an object which stores the value of attributes of a tag and work similar to the HTML attributes. To send props into a component, use the same syntax as HTML attributes:

```
import React from 'react';
import ReactDOM from 'react-dom/client';
 function Car(props) { return <h2>I am a { props.brand }! </h2>;}
const myElement = <Car brand="Ford" />;
const root = ReactDOM.createRoot(document.getElementById('root'));
 root.render(myElement);
```

5 b. Create a spring boot application to maintain Employee details like employee id, employee name & salary in Empoyee.java – an entity class or data JPA with Employee information. Design a Rest Controller with GET, PUT, POST and DELETE Restful APIs. 10 marks

Employee Entity Class – Employee.java

```
import javax.persistence.*;

@Entity          // Annotation
 public class Employee {
        @Id     //Annotation
        private int id;
        private String name;
         private decimal salary;
        public Employee () {}            //Constructors
public Employee (int id, String name, decimal salary) {
this.id = id;
```

this.name = name;

this.salary = salary: }

public int getId() { Return id; }

public void setId(int id) { this.id = id; }

public String getName(){ return name; }

public void setName(String name) { this.name = name;}

public decimal getSalary() { return salary; }

public void setSalary(decimal salary) { this. salary = salary; }

}

Employee Rest Controller Class – EmployeeController.java

@Restontroller //Controller Annotation

public class EmployeeController{

@Autowired //Dependency Injection

private EmployeeService employeeservice; //Service Class Object

@GetMapping("/employees") //Display Employee List

public List<Student> getAllEmployeeDetails(){ return employeeservice.getAllEmployeeDetails();}

@GetMapping("/employee/{id}") //Find Employee by Id

Public List<Employee> getEmployeeDetails(@PathVariable int id){ return employeeservice.getEmployeeDetails(id).orElsethrow();}

@PostMapping("/employees/") //Add Employee Details

public List<Employee> addEmployeeDetails(@RequestBody Employee employee){ employeeservice.addEmployeeDetails(employee);}

@PutMapping("/employees/{id}") //Update Employee Details

public List< Employee > updateEmployeeDetails(@RequestBody Employee employee, @PathVariable int id){

Employeeservice.updateEmployeeDetails(employee,id);}

@DeleteMapping("/employee/") //Delete Employee Details

Public List<Employee> deleteEmployeeDetails(@RequestBody Employee employee){ employeeservice.deleteEmployeeDetails(employee);}

}

6 a) Differentiate spring and spring boot. 10 marks

S.No.	Spring	Spring Boot
1.	Spring is an open-source lightweight framework widely used to develop enterprise applications.	Spring Boot is built on top of the conventional spring framework, widely used to develop REST APIs.
2.	The most important feature of the Spring Framework is dependency injection.	The most important feature of the Spring Boot is Autoconfiguration.
3.	It helps to create a loosely coupled application.	It helps to create a stand-alone application.
4.	To run the Spring application, we need to set the server explicitly.	Spring Boot provides embedded servers such as Tomcat and Jetty etc.
5.	To run the Spring application, a deployment descriptor is required.	There is no requirement for a deployment descriptor.
6.	To create a Spring application, the developers write lots of code.	It reduces the lines of code.
7.	It doesn't provide support for the in-memory database.	It provides support for the in-memory database such as H2.
8.	Developers need to write boilerplate code for smaller tasks.	In Spring Boot, there is reduction in boilerplate code.
9.	Developers have to define dependencies manually in the pom.xml file.	pom.xml file internally handles the required dependencies
10	Its goal is to make Java enterprise edition development easier, allowing developers to be more productive	10. It provides the RAD (Rapid Application Development) feature to the spring framework for faster application development.

6b) Implement programmatically navigation between different components using react Router. 10marks

• App.js

```
import "./App.css";
import { BrowserRouter, Route, Routes } from "react-router-dom";
import Home from "./Home";
```

```
import About from "./About";
import Contact from "./Contact";
import Navbar from "./Navbar";
function App() {
return ( <div className="App">
<BrowserRouter> <Navbar />
<Routes>
<Route path="/" element={<Home />} />
<Route path="/about" element={<About />} />
<Route path="/contact" element={<Contact />} />
</Routes> </BrowserRouter>
</div> );
}; export default App;
```

• Navbar.js

```
import React from "react";
import { Link } from "react-router-dom";
const Navbar = () => {
return ( <div> <nav>
<Link to="/">HOME</Link>
<Link to="/about">ABOUT</Link>
<Link to="/contact">CONTACT</Link>
</nav>
</div>);
}; export default Navbar;
```

• Home.js

```
import React from 'react'const Home = () => { return (
<div><h2>Welcome to Home page</h2></div>
 ); };
export default Home;
```

• About.js

```
import React from "react";const About = () => { return (
 <div> <h2>This is ABOUT page</h2> </div>
); }; export default About;
```

• Contact.js

```
import React from 'react'const Contact = () => { return (
<div><h2>This is CONTACT page</h2></div>
); }; export default Contact;
```

Section-IV

7a. Describe different spring annotations along with their usage and syntax. 10

• **@Entity:** Specifies that the class is an entity. This annotation is applied to the entity class. The entity class is a plain old java object [POJO], contains variables, getter() & setters().

• **@Id:** This annotation is the JPA is used for making specific variable primary key. • **@Autowired**: Spring provides annotation-based auto-wiring by providing @Autowired annotation. It is used to autowire spring bean on setter methods, instance variable, and constructor. When we use @Autowired annotation, the spring container auto-wires the bean by matching data-type.

• **@SpringBootApplication**: This annotation is used to mark the main class of a Spring Boot application. It is a combination of three annotations @EnableAutoConfiguration, @ComponentScan, and @Configuration.

• **@GetMapping**: It maps the HTTP GET requests on the specific handler method. It is used to create a web service endpoint that fetches.

• **@PostMapping**: It maps the HTTP POST requests on the specific handler method. It is used to create a web service endpoint that creates.

• **@PutMapping**: It maps the HTTP PUT requests on the specific handler method. It is used to create a web service endpoint that creates or updates.

• **@DeleteMapping**: It maps the HTTP DELETE requests on the specific handler method. It is used to create a web service endpoint that deletes a resource.

• **@RequestBody**: It is used to bind HTTP request with an object in a method parameter. Internally it uses HTTP MessageConverters to convert the body of the request. When we annotate a method parameter with @RequestBody, the Spring framework binds the incoming HTTP request body to that parameter.

• **@PathVariable**: It is used to extract the values from the URI. It is most suitable for the RESTful web service, where the URL contains a path variable. We can define multiple @PathVariable in a method.

• **@RestController**: It can be considered as a combination of @Controller and @ResponseBody annotations. The @RestController annotation is itself annotated with the @ResponseBody annotation. It eliminates the need for annotating each method with @ResponseBody.

b. Perform the following operations using MongoDB

i. Create a student database.

ii. Insert 2 records into the student collection with suitable fields.

iii. Sort the student by name

iv. Search the student by Id

v. Remove the student by Id 10

i. Create a student database.

In MongoDB, we can create a database using the use command. If the database doesn't exist, MongoDB will create the database when you store any data to it. Using this command, we can also switch from on database to another.

use student // DB Creation

show dbs // To display DB

ii. Insert 2 records into the student collection with suitable fields.

The two documents to student collection can be added using insertMany() method. It is used to insert multiple documents in the collection. The attributes are passed as BSON to insertMany().

db. student.insertMany([{name:”abc”, student_id:123,student_dep:”mno”},

{name:”xyz”, student_id:987,student_dep:”pqr”}]

iii. Sort the student by name

The documents of employee collection is sort by using the find().sort() method. The attribute passed to sort() form the basis for sorting the document. The ascending or descending criteria is mentioned using 1 and -1 respectively.

db.student.find.sort({name:1})

iv. Search the student by Id

The document of student collection is retrieved using the find() method. The attribute passed to find() form the basis for searching the document

db.student.find({id:1})

v. Remove the student by Id A single document of student collection is removed by using the deleteOne() method that satisfy the given criteria.

db.student.deleteOne({id:1})

8a. Illustrate ACID transaction in MongoDB 10

ACID transactions in MongoDB refer to operations that are guaranteed to be Atomic, Consistent, Isolated, and Durable, just like traditional relational databases. Here’s how MongoDB ensures these properties in transactions:

1. Atomicity

Atomicity means that a transaction will either complete entirely or have no effect at all. In MongoDB, a transaction is atomic at the document level, and with multi-document transactions, MongoDB ensures that either all operations in the transaction are applied, or none of them are.

2. Consistency

Consistency ensures that a transaction brings the database from one valid state to another valid state. MongoDB ensures consistency through its transaction mechanisms by ensuring that all operations within a transaction are applied together.

For example in Bank database, the consistency is maintained by ensuring that the sum of the balances in the `accounts` collection remains the same before and after the transaction.

3. Isolation

Isolation ensures that concurrent transactions do not interfere with each other. MongoDB achieves isolation in transactions by using a snapshot of the data that is consistent throughout the transaction.

For example in bank database, even if other transactions are happening simultaneously, MongoDB ensures that the transaction operations are isolated from others until it is committed.

4. Durability

Durability ensures that once a transaction has been committed, it will remain committed, even in the event of a system crash. MongoDB writes transaction logs to ensure that committed data is preserved.

After calling `commitTransaction()`, the changes are durable, and in case of a failure, MongoDB will recover to the last committed state.

By following these principles, MongoDB ensures that its transactions are reliable and adhere to ACID properties, making it a robust choice for applications requiring strong consistency guarantees.

b. Demonstrate with a simple code to secure REST APIs with Spring Security. 10marks

Spring Security is an application-level security framework which provides various security features like: authentication, authorization to create secure Java Enterprise Applications. To enable application-level security feature, add spring security dependency to the pom.xml or you can add while creating a maven spring boot application using spring initilizr. Spring security make sure that each and every API written in the controller to be authenticated.

The framework targets two major areas of application are authentication and authorization.

• Authentication is the process of knowing and identifying the user that wants to access.

• Authorization is the process to allow authority to perform actions in the application. The application-level framework features provide Login and logout functionality. It allow/block access to URLs to logged in users. It also allow/block access to URLs to logged in users AND with certain roles.

• pox.xml code for spring security dependency

```
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-security</artifactId>
</dependency>
```

• SecurityController.java class

```
@RestController
public class SecurityController {
@GetMapping("/")
public String Welcome() {
return ("<h1>Welcome to SpringBoot Security</h1>");
} }
```

A default system generated security password is given by the spring boot security framework. The password is dynamic and changes every time we execute the application. Try to access the REST API using the URI mapping that is http://localhost:8080/ in any browser or application like postman or swagger. As the URI hits in the browser a default login form authentication is enabled. By default, User is created as username for authentication. Spring security also provide logout form and error handling / validation.

We can make the password static by setting its value in the application.properties file. Application.properties file is created under src/main/resource package. We can also create static user and static password by setting its value in the application.properties file as shown below.

spring.security.user.name=ABC

spring.security.user.password=XYZ

Section V

9. a) Differentiate automated and manual deployment process. -10

Characteristics	Manual Deployment	Automated Deployment
Process and Human Intervention	Manual Deployment approach, deployment tasks are performed manually by human operators. This involves activities like copying files, configuring settings, and executing scripts step by step. Each deployment action requires explicit human input and decision-making.	Automated deployment relies on software tools, scripts, and systems to carry out the deployment process. Developers and operations teams set up deployment pipelines that automatically execute predefined steps, reducing the need for direct human intervention.
Speed and Efficiency	Manual Deployment: Because manual deployment involves human operators performing tasks, it can be slow and prone to errors. The time taken to deploy applications may vary based on individual skill levels and the complexity of the application.	Automated Deployment: Automated deployment is generally much faster and more efficient. Once the deployment pipeline is set up, it can be executed repeatedly without human intervention. This consistency ensures that the deployment process is reliable and predictable.
Consistency and Reproducibility	Human-based deployment can lead to inconsistencies between different environments (e.g., development, testing, production). If the same steps are not followed accurately, issues may arise during deployment.	Automated deployment ensures consistency and reproducibility. The same deployment process is applied across all environments, reducing the risk of configuration drift and errors.
Risk and Error Minimization	Human error is a significant risk in manual deployment. Typos, mis-configurations, or missed steps can lead to deployment failures and downtime.	Automated systems can help minimize human errors as the deployment process is predefined and thoroughly tested. Automated rollback mechanisms can also be implemented to revert to a stable state in case of deployment issues.
Scalability and Complexity	Manual deployment becomes challenging and time-consuming as the application and infrastructure scale in size and complexity.	Automation can handle more complex deployments and scale easily as it is designed to handle a wide range of scenarios. It is well-suited for modern microservices-based architectures and cloud-native applications.
Continuous Deployment & Continuous Integration (CI/CD)	Implementing continuous deployment and continuous integration without automation is difficult and may not be practical.	Automated deployment is a key enabler for CI/CD workflows, where changes are automatically tested, integrated, and deployed to production, often multiple times a day.

9 b) Discuss Components of application performance management (APM).- 10marks

The five most important elements of application performance management are

1. End-user experience monitoring,
2. Runtime application architecture discovery,
3. User-defined transaction profiling,
4. Component deep-dive monitoring, and
5. Analytics

1. End-User Experience Monitoring

End-user experience monitoring tools gather information on the user's interaction with the application and help identify any problems that are having a negative impact on the end-user's experience.

It's important tool that can monitor both on-premises and hosted applications. It's also helpful to consider a tool that allows for instant changes to network links or external servers if either or both of these are compromising the end-user experience.

2. Runtime Application Architecture Discovery

The hardware and software components involved in application execution as well as the paths they use to communicate to help pinpoint problems and establish their scope.

With the complexity of today's networks, discovering and displaying all the components that contribute to application performance is a hefty task. It is a monitoring tool that provides real-time insight into the application delivery infrastructure. It will also allow the IT team to interact with different aspects of the APM solution quickly, efficiently, and effectively.

3. User-Defined Transaction Profiling

It is important to identify the software, hardware, and communication path that application traffic takes throughout the networks, it is equally important to understand user-defined transactions as they traverse the architecture.

This third step will help ensure two things. First, it will allow federal IT pros to trace events as they occur across the various components. Second, it will provide an understanding of where and when events are occurring, and whether they are occurring as efficiently as possible.

4. Component Deep-Dive Monitoring

This step provides an in-depth understanding of the components and pathways discovered in previous steps. In a nutshell, the federal IT pro conducts in-depth monitoring of the resources used by, and events occurring within, the application performance infrastructure.

5. Analytics

Finally, as with any IT scenario, having information is one thing; understanding it is another. APM analytics tools allow federal IT pros to:

 Set a performance baseline that provides an understanding of current and historical performance, and set an expectation of what a "normal" application workload is

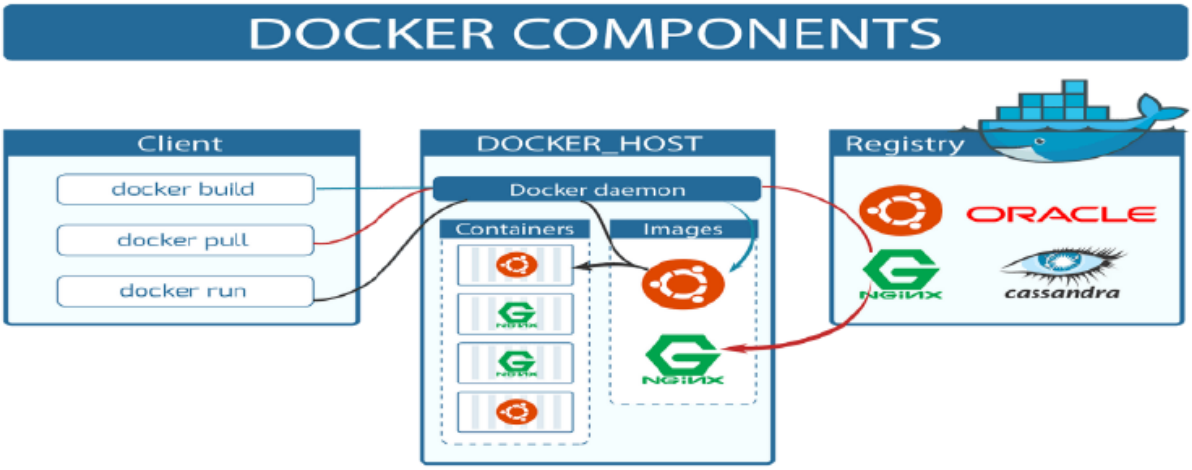
 Quickly identify, pinpoint, and eliminate application performance issues based on historical/baseline data.

 Anticipate and mitigate potential future issues through actionable patterns.

 Identify areas for improvement by mapping infrastructure changes to performance changes.

10. a) Discuss the Components of Docker Container. 10 marks

Docker is an open-source software platform. It is designed to make it easier to create, deploy, and run applications by using containers. Containers allow a developer to package up an application with all of the parts which are required, such as libraries and other dependencies and ship it all out as one package. The Docker Components are



• Docker client

The Docker client enables users to interact with Docker. Docker runs in a client-server architecture that means docker client can connect to the docker host locally or Source Code remotely. Docker client and host (daemon) can run on the same host or can run on different hosts and communicate through sockets or a RESTful API.

The Docker client is the primary way that many Docker users interact with Docker. When we use commands such as docker run, the client sends these commands to docker daemon, which carries them out. The docker command uses the Docker API. The Docker client can communicate with more than one daemon. We can communicate with the docker client using the Docker CLI with commands. Then the docker client passes those commands to the Docker daemon. Commands are docker build, docker run, docker push, etc.

• Docker Host

The Docker host provides a complete environment to execute and run applications. It includes Docker daemon, Images, Containers, Networks, and Storage. a) Docker Daemon: A persistent background process that manages Docker images, containers, networks and storage volumes. The Docker daemon constantly listens for Docker API requests and processes them. b) Docker Images: A read-only binary template used to build containers. It contains metadata that describe the container's capabilities and needs. An image can be used to build a container. Container images can be shared across teams using a private container registry, or shared with the world using a public registry like Docker Hub. c) Docker Containers: A runnable instance of an image. We can create, start, stop, move, or delete a container using the Docker API or CLI. We can connect a container to one or more networks, attach storage to it, or even create a new image based on its current state. d) Docker Networking: We can communicate one container to other containers. The default networks are – none, bridge, and host. The none and host networks are part of the network stack in Docker. The bridge network automatically creates a gateway and IP subnet and all containers that belong to this network can talk to each other via IP addressing. e) Docker Storage: A container is volatile it means whenever we remove or kill the container then all of its data will be lost from it. Docker offers Data Volumes, Data-Volume Container, Bind Mounts

• Docker Registries

Docker-registries are services that provide locations from where we can store and download images. A Docker registry contains repositories that host one or more Docker Images. Public

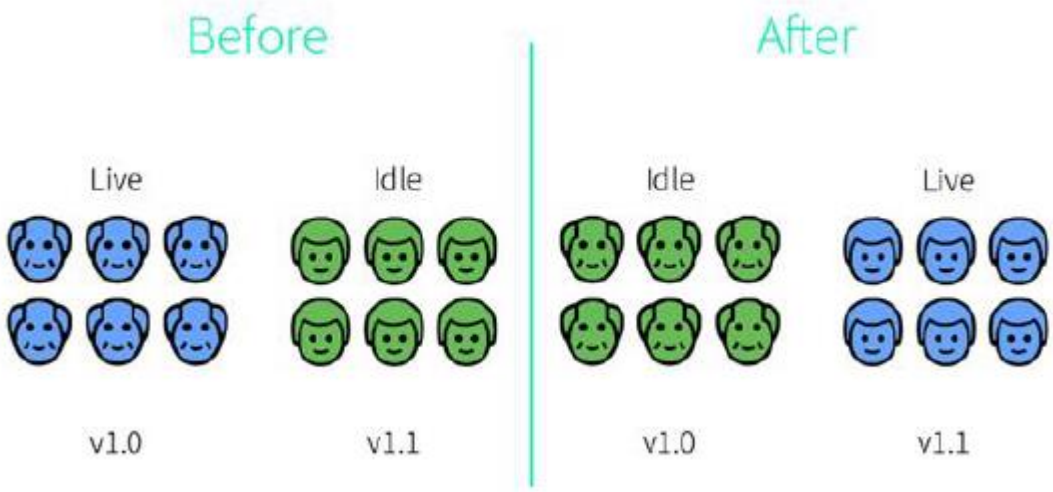
Registries include Docker Hub and Docker Cloud and private Registries can also be used. We can also create our own private registry. Push or pull image from docker registry using the commands docker push docker pull docker run.

10 b. Illustrate the working Blue-Green and Canary deployment strategies with neat diagram. 10

• **Blue/green deployment** Blue/green deployment is a deployment technique to release new code into the production environment. Blue/green deployments make use of two identical production environments – known as Blue which is active and the other is Green which is set to idle. New updates are pushed to the active environment where it is monitored for bugs while the idle environment serves as a backup where traffic can be routed in case an error occurs.

The primary goal of blue-green deployment is to minimize downtime and reduce the risk of introducing new software versions by allowing you to switch between these environments seamlessly.

When you deploy a new version of your application, you do so in the inactive environment (say, Green). This allows you to test the new version thoroughly, ensure that it's stable, and make any necessary adjustments without affecting your users. Once you're confident that the new version is reliable, you can switch users to the Green environment, making it the new active production environment. The previously active Blue environment now becomes the standby, ready to receive the next deployment.

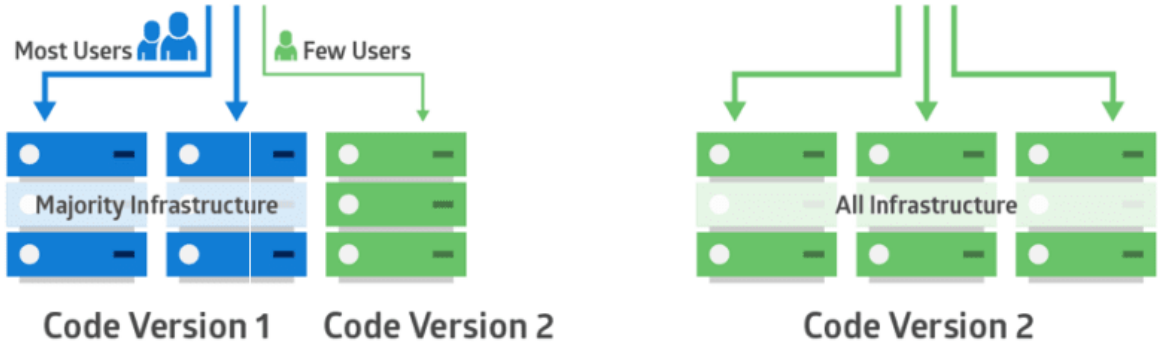


• **Canary deployment** Canary deployment is a technique to reduce the risk of updating software or introducing new changes in the production environment by slowly rolling out the change to a small subset of users before making the software functional for everyone.

This method helps you identify and address any potential issues with the new release before it affects all users, thus minimizing the risk of a widespread problem or outage.

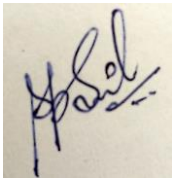
In canary deployment, you gradually roll out the new version of your application to a percentage of users, monitor their feedback and performance, and iterate on any issues that may arise. If the new version performs well and receives positive feedback, you can proceed with a full-scale release. On the other hand, if you encounter any problems, you can pause the rollout, address the issues, and then continue the process. This incremental approach helps

ensure that any potential problems are detected and resolved early on, reducing the risk of a catastrophic failure.



CERTIFICATE

Certified that, as per the guidelines the question paper and the model answers are prepared and typed by me for the course **Full Stack Development-20CS52I** and they are found correct according to my knowledge.



Sunil.B.P
Sr. Scale Lecturer, CSE
Govt.Polytechnic Karkala
Id: 137476