**V Semester Diploma Examination, December-2023**

**Full Stack Development-20CS52I**

**Scheme of Valuation**

**SECTION I**

1a) Definition: 1 Mark, Benefits: 4 Marks, Explanation with example: 5 Marks

1b) Explanation -10 Marks

2a) Explanation of 5 different processes=5x2=10 Marks

2b) Identification=2 Marks, Characteristics=4 Marks, Advantages=4 Marks

**SECTION II**

3a) Writing of any 4 user stories with Acceptance criteria=4x3=12 Marks

3b) Writing any 4 test cases=4x2=8 Marks

4a) Writing of any 4 user stories with Acceptance criteria=4x3=12 Marks

4b) Writing any 4 test cases=4x2=8 Marks

**SECTION III**

5a) Illustration of each sub question=2 Marks. 5x2=10 Marks

5b) Explanation about bulkWrite() method=**2 Marks**, Listing of various bulkWrite( ) methods=**2 Marks**, Explanation about any 2 bulkWrite() methods=2x3=**6 Marks**

6a) Explaining States=5 Marks, Explaining Props=5 Marks

6b) Listing of 10 differences= 10 Marks

**SECTION IV**

7a) Listing Any 10 differences=10 Marks

7b) Designing Entity class=5 Marks, Controller class=5 Marks

8a) Class creation= 2 Marks, writing beans.xml=4 Marks, Writing main class with Application Context=4 Marks

8b) Explaining any 5 rules= 5x2=10 Marks

**SECTION V**

9a) Diagram=4 Marks, Listing and explaining all 6 components=6 Marks

9b) Diagram=4 Marks, Explaining 6 phases= 6 Marks

10a) Identification of the Canary deployment strategy=2 Marks, Diagram= 3 Marks, Explanation= 5 Marks

10b) Definition of Kubernetes= 3 Marks, 7 Automation steps=7 Marks.

**Model Answers**

**Section-I**

1. **a) Explain how digital transformation has brought revolution in retail purchases with an example. 10 Marks**

Digital transformation is transforming the relationship between the customers and various brands. Retail digital transformation is including the technology in the retail industry to enhance operational efficiency, improve customer experience and drive growth.

According to the latest research, more than 50% of all consumer spending will be done online, and almost 60% of it will be done through mobile devices. One of the key advantages of digital transformation in retail is the ability to deliver personalized customer interactions. By collecting and analyzing customer data, retailers can gain valuable insights into individual preferences, purchase history, and browsing behaviour.

**Benefits:**

1. Increased customer retention
2. Providing useful market insights
3. Easier inventory management
4. Implementing efficient marketing campaigns
5. Improved customer service
6. Increased sales
7. Reduced Inefficiencies.

**Example for one such digital retailing is Amazon shopping application software**

* Using this application, an user can shop various products without having to visit the store.
* User gets to compare the products of various brands and choose the best. Among them
* The user gets an easy refund or exchange of the product when compared to traditional shopping. An user can return back the product after delivery if he/she is not satisfied with the quality or some other issues.
* The application supports different modes of secure payment such as cash on delivery, net banking, UPI payment etc making it a hassle free for the customers.
* The application helps the users to gift a product to their loved ones as it allows shipping of the product to our desired address.
* The application satisfies the customers by offering beautiful personalized products.
* The application helps the customers to track their products before delivery.
* The digital transformation in retail purchases has made it easy for the people to shop a wide variety of products sitting at home.

**1 b) Diagnostic imaging procedures are unpleasant for patients. Explain how design thinking helped Doug Dietz, an industrial designer to create a scanner experience that children loved. 10 Marks**

After Doug Dietz, an Industrial Engineer working for GE Healthcare developed an MRI Scanner, he paid a visit to his product as he was very excited to finally see it in action. However, what he encountered in the hospital shocked him. The device giving him such pride put children through their worst fears. Doug Dietz noticed a girl crying before undertaking the MRI scan and her stressed-out, worried parents who had to put their child through this and watch helplessly.

With the perspective of this child, Doug Dietz noticed, that the whole room was cold, sterile, and dark. He also found out, that 80% of the children undergoing an MRI Scan need to be sedated so that they keep still and the scan can be used in the end. This information combined with the emotions he noticed in the hospital, inspired him to redesign his scanners and develop children-friendly rooms for medical scanners like MRIS, CTS and X-Rays.

Doug Dietz decided to use design thinking to find inspiration for his project as it is very human-centered. His goal was to look at everything through the eyes of a child and understand their needs. To really understand how children think and perceive their environment, he performed several activities. First, he visited a day care center to observe children’s behavior and gain empathy for them and their way of thinking. He then observed children undergoing procedures and consulted doctors and child life specialists. With the gained information and more input from talks with other managers and users of existing products, he started an iterative process of improving his ideas, resulting in a first prototype.

This prototype was installed in a children’s hospital. The scanners were placed an into adventure-themed rooms. The idea is to let children immerse in a world of adventure as soon as they enter the room. There were rocks on the floor on the way to the scanner. Then there is a waterfall cascading from the wall that is floating into a pond, in which the scanner is positioned. The scanner itself is designed as a canoe that lowers down into the pond when the children hop onto it. The children are told that if they keep still, the fish will start jumping around them.

**2 a) An enterprise is a new startup for a business organization. Analyse how different processes are involved in 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.

1. **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.

1. **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.

1. **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.

1. **Coordinating Activities**

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

**2 b) Identify the following cloud service types and list their characteristics and advantages: Cisco WebEx, Google App Engine, Amazon EC2 10 Marks**

1. **Cisco WebEx –**

It is software-as-a-service (SaaS) type of cloud service.

**Characteristics of Cisco WebEx**

• High-Quality Video Conferencing

• Cross-Platform, Functional and Geographic Versatility

• File and Desktop Sharing

• Brainstorming via Whiteboarding

• AI Powered Functionality

• Security

**Advantages Of Webex**

• Easy Sharing Option

• Easy Invitation

• Difficult To Share Confidential Data

• HD Video And Audio

• Minimum Utilization Of Internet Data

• Accessible To Various Tools

• Sharing Task Is Easy

• Best Platform For Education Sector

• Stream Live Meetings On Social Media

• Reasonable Subscription Plans

1. **Google App Engine-**

It is Platform as a Service (PaaS) type of cloud service.

**Characteristics of Google App Engine**

• Popular language

• Open and flexible

• Fully managed

• Powerful application diagnostics

• Application versioning

• Application security

**Advantages Of Google App Engine**

• Open and familiar languages and tools

• Just add code

• Pay only for what you use

• Easy to build and use the platform:

• Scalability

• Various API sets

1. **Amazon EC2-**

It belongs to IaaS Cloud Computing Services

**Characteristics of Amazon EC2**

• Safe

• Dynamic Scalability

• Full Control of Instances

• Configuration Flexibility

• Integration with Other Amazon Web Services

• Reliable and Resilient Performance Amazon Elastic Block Store (EBS)

• Support for Use in Geographically Disparate Locations

• Cost Effective

• Credible

• Flexibility of Tools

• Created for Amazon Web Services

**Advantages of Amazon EC2**

• Reliability

• Security

• Flexibility

• Cost Savings

• Complete Computing Solution

• Elastic Web Computing

• Complete Controlled setup

**Section-II**

**3 a) BookingHall is an online convention hall booking application which allows users to login and find the halls in a particular locality and check the availability of a hall for specific dates. Users can block a hall for required duration and get booking details. Identify and write the user stories for this application. 12 Marks**

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

For BookingHall 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 hall booking app.

**Acceptance Criteria:**

1. While signing up, enter Username, Email, Password & Confirm Password, Security question and Address.
2. If user sign up with an incorrect detail, user receives an error message for incorrect information.
3. If user tries to sign up with an existing email address, existing mobile number, user receives error messages saying "email exists", “mobile registered already”.
4. 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
5. **User Story: Login**

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

**Acceptance Criteria:**

1. Username, password and captcha are required for user login.
2. If we are trying to login with incorrect username or password, then error message will be displayed as "bad credentials”.
3. After successful log in, home page is displayed.
4. **User Story: Search**

As an authorized user, I want to search the halls in a particular locality for a particular date so that I can book the hall.

**Acceptance Criteria:**

1. User has to enter valid locality and valid date.
2. A valid search displays list of halls available in a particular locality for a particular date.
3. User can sort, filter and modify the search results.
4. **User Story: Book Hall**

As an authorized user, I want to book a hall so that I can conduct an event at a particular place on a date.

**Acceptance Criteria:**

* User has to choose among the available halls 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 booking details to registered mobile number and E-mail id

1. **User Stories: Logout**

As a logged in user, I want to log out of BookingHall app so that I can prevent unauthorized access to my profile.

**Acceptance Criteria:**

1. 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.
2. 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. 8 Marks**

**Test Cases for the Registration Page:**

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

**Test Cases for the Login Page:**

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

**Test Cases for the Hall Search:**

1. Ensure that the Hall search page displays list of halls corresponding to given particular date and locality.
2. Verify whether user is able to apply filter, sort halls and modify existing search.
3. Verify whether user is able to select the required hall, check for availability of hall for a particular date and locality.

**Test Cases for the Hall Booking:**

1. Verify that the system displays the total cost of the hall as per booking details, including any taxes and fees.
2. Verify whether user is able to cancel the hall or not.
3. Verify whether booking confirmation is received by user or not.

**Test Cases for the Payment Gateway:**

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

**4 a) eDesert is an online application which allows users to login and search for a desert, sort the desert list based on the rating or price, select the items, add them to cart and place order by doing payment. Identify and write the user stories for this application.**

**12 Marks**

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

For eDesert 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 desert booking 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

1. **User Story: Login**

As a registered/authorized user, I want to login into eDesert application so that I can view and book a variety of deserts provided by 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 "bad credentials”.
* After successful log in, home page is displayed.

1. **User Story: Search**

As an authorized user, I want to search the variety of deserts so that I can book the desert of my choice.

**Acceptance Criteria:**

* User has to enter valid desert name.
* A valid search displays list of deserts available
* User can sort, filter and modify the search results based on the rating or price.

1. **User Story: Desert booking**

As an authorized user, I want to book a desert of my choice from the list of available deserts

**Acceptance Criteria:**

* User has to choose among the available deserts 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 booking details to registered mobile

number and E-mail id

1. **User Stories: Logout**

As a logged in user, I want to log out of eDesert app so that I can prevent unauthorized access to my 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. 8 Marks**

**Test Cases for the Registration Page:**

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

**Test Cases for the Login Page:**

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

**Test Cases for the Desert Search:**

1. Ensure that the Desert search page displays list of deserts.
2. Verify whether user is able to apply filter, sort deserts and modify existing search.
3. Verify whether user is able to select the required desert, check for its availability and quantity.

**Test Cases for the Desert Booking:**

1. Verify that the system displays the total cost of the desert as per booking details, including any taxes and fees.
2. Verify whether user is able to cancel the deserts or not.
3. Verify whether booking confirmation is received by user or not.

**Test Cases for the Payment Gateway:**

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

**Section-III**

**5 a) An organization needs to maintain its employee details in database. Illustrate how MongoDB is very effective in 10 Marks**

1. **Creating and dropping database.**
2. **Creating and dropping collections**
3. **Creating user and role**
4. **Removing an employee detail based on employee id**
5. **Creating 5 documents in employee collections.**
6. **Creating the database**

Syntax: use db\_name

Example: >use myprojectdb

switched to db myprojectdb

Here 'use' is the command for creating a new database in MongoDB and 'myprojectdb' is the name of the database.

**Dropping the database**

We can drop the database through **db.dropDatabase()** method. It also removes the current database, deleting all associated files.

We need to connect to mongoDb database by giving the command **mongo**. After that we can check all the databases with **show dbs** command.

We need to go inside the database that we are dropping, by using **use <dbName>** command. Then the database can be dropped using **db.dropDatabase()** method

Example: >mongoDb

>show dbs

> use myprojectdb

>db.dropDatabase( )

1. **Creating the collections**

In MongoDB, a new collection is created when we add one or more documents to it. We can insert documents in the collection using the following methods:

Example: db.myNewCollection1.insertOne( { name:"geeksforgeeks" } )

Here, we create a collection named as myNewCollection1 by inserting a document that contains a name field with its value in it using [insertOne()](https://www.geeksforgeeks.org/mongodb-insert-single-document-using-mongoshell/) method.

**Dropping the collections:**

db.collection.drop( ) method removes a collection from the database. The method also removes any indexes associated with the dropped collection.

Example: To drop a collection “users” of database myprojectdb:

> use myprojectdb

> db.users.drop()

1. **Creating user and role**

In MongoDB, we can also create a user for single database using createUser() method.

**Example:**

db.createUser(

{

user: "robert",

pwd: "hellojose",

roles:[{role: "userAdmin" , db:"example"}]})

Here, we create a user whose user name is “Robert”, password is “hellojose”, and we assign a role for the user which in this case needs to be a database administrator so it is assigned to the “userAdmin” role. This role will allow the user to have administrative privileges only to the database specified in the db option

1. **Removing an employee detail based on employee id**

A single document of employee collection is removed by using the deleteOne() method that satisfies the given criteria.

Example: db.employee.deleteOne({empid:123})

1. **Creating 5 documents in employee collections.**

Five documents to employee 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().

Example:

db.employee.insertMany( [ {name:”abc”, emp\_id:123,emp\_dep:”mno”},

{name:”xyz”, emp\_id:987,emp\_dep:”pqr”},

{name:”rtc”, emp\_id:253,emp\_dep:”erf”},

{name:”nko”, emp\_id:756,emp\_dep:”klr”},

{name:”tyj”, emp\_id:659,emp\_dep:”wer”} ] )

**5 b) The write operations on the collections are very high. Explain the technic applicable to manage the scenario 10 Marks**

When the write operations on the collections are very high, it is a good practice to use db.collection.bulkwrite( ) method to handle the repeated write operations on the collections.

The [db.collection.bulkWrite()](https://www.mongodb.com/docs/manual/reference/method/db.collection.bulkWrite/" \l "mongodb-method-db.collection.bulkWrite) method provides the ability to perform bulk insert, update, and delete operations.

[bulkWrite()](https://www.mongodb.com/docs/manual/reference/method/db.collection.bulkWrite/#mongodb-method-db.collection.bulkWrite) supports the following write operations:

* [insertOne](https://www.mongodb.com/docs/manual/reference/method/db.collection.bulkWrite/#std-label-bulkwrite-write-operations-insertOne)
* [updateOne](https://www.mongodb.com/docs/manual/reference/method/db.collection.bulkWrite/#std-label-bulkwrite-write-operations-updateOneMany)
* [updateMany](https://www.mongodb.com/docs/manual/reference/method/db.collection.bulkWrite/#std-label-bulkwrite-write-operations-updateOneMany)
* [replaceOne](https://www.mongodb.com/docs/manual/reference/method/db.collection.bulkWrite/#std-label-bulkwrite-write-operations-replaceOne)
* [deleteOne](https://www.mongodb.com/docs/manual/reference/method/db.collection.bulkWrite/#std-label-bulkwrite-write-operations-deleteOneMany)
* [deleteMany](https://www.mongodb.com/docs/manual/reference/method/db.collection.bulkWrite/#std-label-bulkwrite-write-operations-deleteOneMany)

Example:

The example in this section uses the pizzas collection:

|  |
| --- |
| db.pizzas.insertMany( [ |
| { \_id: 0, type: "pepperoni", size: "small", price: 4 }, |
| { \_id: 1, type: "cheese", size: "medium", price: 7 }, |
| { \_id: 2, type: "vegan", size: "large", price: 8 } |
| ] )  The following [bulkWrite()](https://www.mongodb.com/docs/manual/reference/method/db.collection.bulkWrite/" \l "mongodb-method-db.collection.bulkWrite) example runs these operations on the pizzas collection:   * Adds two documents using insertOne. * Updates a document using updateOne. * Deletes a document using deleteOne.  |  | | --- | | db.pizzas.bulkWrite( [ | | { insertOne: { document: { \_id: 3, type: "beef", size: "medium", price: 6 } } }, | | { insertOne: { document: { \_id: 4, type: "sausage", size: "large", price: 10 } } }, | | { updateOne: { | | filter: { type: "cheese" }, | | update: { $set: { price: 8 } } | | } }, | | { deleteOne: { filter: { type: "pepperoni"} } } ] ) | |

1. **a) Discuss 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);

1. **b) Compare the databases MySQL and MongoDB. 10 marks**

|  |
| --- |
| **MySQL MongoDB** |
| 1.It is relational database 1. It is NoSQL database. |
| 2. Suitable for Structured data with clear schema 2. Suitable for Unstructured data |
| 3. Represents data as tables and rows 3. Represents data as JSON docs |
| 4. Must specify tables and columns 4. No need to declare the schema |
| 5. Allows Join operations 5. Doesn’t allow Join operations |
| 6. Utilizes Structured query language 6. Query language is Javascript |
| 7. More suitable when you are just 7. More suitable when you need high |
| starting the database which doesn’t scale much availability Of data with auto data recovery |
| 8.There is no auto failure recovery 8. It supports auto failure recovery |
| 9. It is Slower 9. It is faster than MySQL |
| 10. Works better for smaller datasets 10. Works better for larger datasets |

**Section-IV**

1. **a) Compare 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. |

1. **b) RESTful API supports CRUD operations. Design Entity class and Controller class. 10 marks**

Student Entity Class – Student.java

import javax.persistence.\*;

@Entity // Annotation

public class Student {

@Id //Annotation

private int id;

private String name;

private String branch;

Public Student() {} //Constructors

Public Student(int id, String name, String branch) {

This.id = id;

this.name = name;

this.branch = branch: }

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 String getBranch() { return branch; }

Public void setBranch(String branch) { this.branch = branch; }

}

Student Rest Controller Class – StudentController.java

@Restontroller //Controller Annotation

public class StudentController{

@Autowired //Dependency Injection

private StudentService studentservice; //Service Class Object

@GetMapping(“/students”) //Display Student List

public List<Student> getAllStudentDetails(){ return studentservice.getAllStudentDetails();}

@GetMapping(“/students/{id}”) //Find Student by Id

Public List<Student> getStudentDetails(@PathVariable int id){ return studentservice.getStudentDetails(id).orElsethrow();}

@PostMapping(“/students/”) //Add Student Details

Public List<Student> addStudentDetails(@RequestBody Student student){ studentservice.addStudentDetails(student);}

@PutMapping(“/students/{id}”) //Update Student Details

public List<Student> updateStudentDetails(@RequestBody Student student, @PathVariable int id){

Studentservice.updateStudentDetails(student,id);}

@DeleteMapping(“/students/”) //Delete Student Details

Public List<Student> deleteStudentDetails(@RequestBody Student student){ studentservice.deleteStudentDetails(student);}

}

1. **a) Implement Inversion of Control (loC) container in Spring Boot with an example. -10 marks**

**Implementation:** So now let’s understand what is IoC in Spring with an example. Suppose we have one interface named Sim and it has some abstract methods calling() and data().

|  |
| --- |
| // Java Program to Illustrate Sim Interface  **public** **interface** Sim  {  **void** calling();  **void** data();  } |

Now we have created another two classes Airtel and Jio which implement the Sim interface and override the interface methods.

|  |
| --- |
| // Java Program to Illustrate Airtel Class   // Class  // Implementing Sim interface  **public** **class** Airtel **implements** Sim {       @Override **public** **void** calling()      {          System.out.println("Airtel Calling");      }       @Override **public** **void** data()      {          System.out.println("Airtel Data");      }  } |
| // Java Program to Illustrate Jio Class   // Class  // Implementing Sim interface  **public** **class** Jio **implements** Sim{      @Override  **public** **void** calling() {          System.out.println("Jio Calling");      }       @Override  **public** **void** data() {          System.out.println("Jio Data");      }  } |

**beans.xml**

|  |
| --- |
| <?**xml** version="1.0" encoding="UTF-8"?>  <**beans** xmlns="<http://www.springframework.org/schema/beans>"         xmlns:xsi="<http://www.w3.org/2001/XMLSchema-instance>"         xsi:schemaLocation="<http://www.springframework.org/schema/beans>  <https://www.springframework.org/schema/beans/spring-beans.xsd>">      <**bean** id="sim" class="Jio"></**bean**>    <**bean** id="sim1" class=" Airtel "></**bean**>   </**beans**> |

**Bean Definition**: In Spring, the objects that form the backbone of your application and that are managed by the Spring IoC container are called beans. A bean is an object that is instantiated, assembled, and otherwise managed by a Spring IoC container.

|  |
| --- |
| **import** org.springframework.context.ApplicationContext;  **import** org.springframework.context.support.ClassPathXmlApplicationContext;    **public** **class** Mobile {  **public** **static** **void** main(String[] args) {          // Using ApplicationContext tom implement Spring IoC          ApplicationContext applicationContext = **new** ClassPathXmlApplicationContext("beans.xml");           // Get the bean          Sim sim = applicationContext.getBean("sim", Sim.**class**);          Sim sim1 = applicationContext.getBean("sim1", Sim.**class**);          // Calling the methods          sim.calling();          sim.data();  sim1.calling();          sim1.data();      }  } |

**Output**

Jio Calling

Jio Data

Airtel Calling

Airtel Data

1. **b)** **Explain the rules to follow for an API to be RESTful. -10 marks**

For an API to be RESTful there are six rules that it needs to follow. The rules are as follows:

* 1. Uniform interface
  2. Client–server
  3. Stateless
  4. Cacheable
  5. Layered system
  6. Code on demand

1. **Uniform Interface**

The API should facilitate communication between the client and server as they exchange data. To efficiently exchange data, we need a uniform interface.

If our system is using well known protocols and techniques, it’s easily implemented. Data should be exchanged using standard formats JSON or XML.

1. **Client-server architecture**

The main purchase of an API is to connect two pieces of software – software might be custom built and run, off the shelf, or a Software as a Service. The client makes requests and the server gives responses – it’s important that they stay separate and independent.

1. **Stateless**

It’s important that each endpoint in the API be stateless which means each call must be handled independently and have no knowledge of what happened from other calls.

A stateless API means that the server receives everything from a client that they need to identify them and what they want in each request.

The major advantages of a stateless API are:

They can handle more clients because less resources are used, and each request is independent of previous ones.

1. **Cacheable**

APIs can have a lot of overhead when they process requests – making repeated requests for data that rarely changes or for the exact same data doesn’t normally make sense.

A cache allows us to temporarily store data locally for a agreed upon period of time. So essentially, if the client goes to make the call again and the agreed upon time hasn’t been fully spent it will use the stored version.

1. **Layered System**

REST allows us to build a layered system architecture meaning that multiple servers may potentially respond to a request. A client shouldn’t be able to easily tell what system is responding to their request especially if it’s behind an API Gateway.

1. **Code on Demand**

Code-on-Demand (COD) is the only optional constraint in REST. It allows clients to improve its flexibility because, in fact, it is the server who decides how certain things will be done. For example, client can download a JavaScript, java applet or even a flash application in order to encrypt communication so servers are not aware of any encryption routines / keys used in this process.

**Section V**

**9 a) Discuss the components of docker container with neat diagram 10 Marks**

**Components of Docker:-**

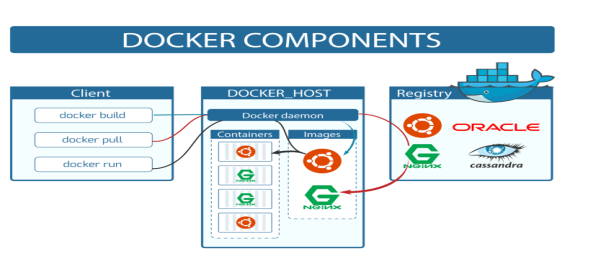
1. Docker container

2. Docker client

3. Docker daemon

4. Docker image

5. Docker registry



* **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 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. When you use commands such as docker run, the client sends these commands to docker daemon, which carries them out.

* **Docker Host**

The Docker host provides a complete environment to execute and run applications. It includes Docker daemon, Images, Containers, Networks, and Storage.

* **Docker Daemon**

Docker Daemon is 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. When you use a docker run command to start up a container. Your docker client will translate that command into an HTTP API call, sends it to docker daemon. Docker daemon then evaluates the request, talks to underlying os and provisions your container

* **Docker Images:**

Docker-images are a read-only binary template used to build containers. Images also contain metadata that describe the container’s capabilities and needs. Create a docker image using the docker build command. whenever you pass a Dockerfile to the docker build command then the docker daemon will create a docker image according to the Dockerfile instruction. Run the docker images using the docker run command. whenever we pass the command to docker client then the docker client passes this command to the docker daemon then docker daemon will create the container for that image. Push the docker image to the public registry like DockerHub using the docker push command after pushed you can access these images from anywhere using docker pull command.

* **Docker Containers:**

A container is a runnable instance of an image. You can create, start, stop, move, or delete a container using the Docker API or CLI. Containers provide you with a lightweight and platform-independent way of running your applications. A container is volatile it means whenever you remove or kill the container then all of its data will be lost from it. If you want to persist the container data use the docker storage concept. Containers only have access to resources that are defined in the image, unless additional access is defined when building the image into a container.

* **Docker Registries**

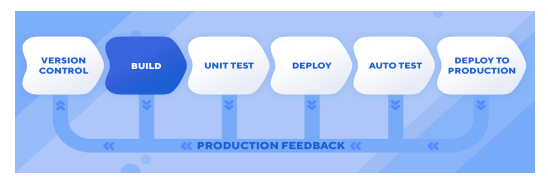
Docker-registries are services that provide locations from where you can store and download images. In other words, a Docker registry contains repositories that host one or more Docker Images. Public Registries include Docker Hub and Docker Cloud and private Registries can also be used. You can also create your own private registry.

**9 b) Draw a process flow diagram for Continuous Integration CI / Continuous Delivery CD of any online application and explain each component. 10 Marks**

A typical CI/CD pipeline must include these phases:

* Version Control
* Build phase
* Testing phase
* Deploy phase
* Automated testing phase
* Deploy to the production phase

In such a pipeline, things started with the developer team to write the initial lines of code. The developers then commit these codes into a version control system, which is the first phase of the pipeline. As a linear workflow, the developers will commit new codes and push them to the version control system with an updated version tag.

****

**The build phase**

The build phase is triggered when new codes are pushed to a repository. Because the initial codes are stored in small branches of the repository, the compiler will gather all features of the codes and their dependencies, then compile them into a new build.

**The testing phase**

The testing phase comprises of multiple types of tests, with the most crucial one being unit testing. Unit testing will test the individual units of the product from its source code

**Deploy phase**

Once the builds have passed the tests, they are moved to the deployment phase then pushed into a test server. This phase allows developers to simulate the product in a production-equivalent environment to examine see the product features.

**The automation test phase**

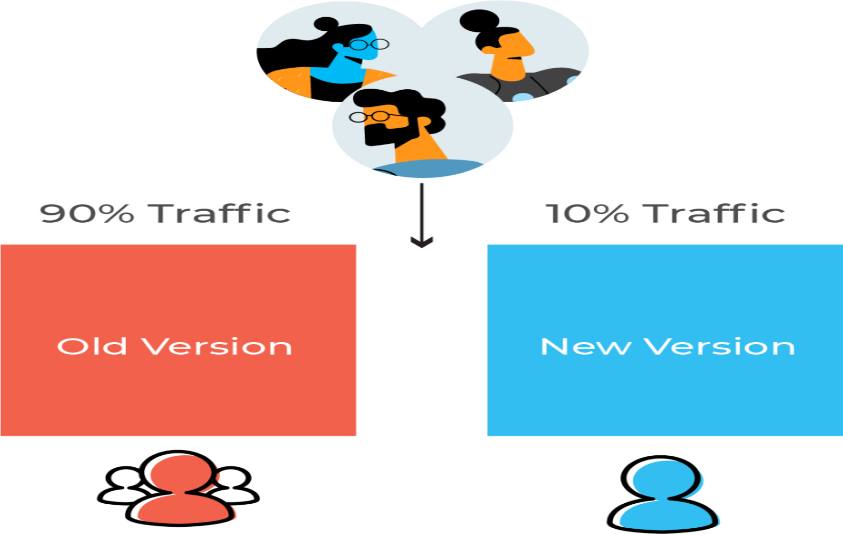
It will perform the final tests to qualify the built features before they are deployed to production. Automated and continuous testing is applied in this phase to utilize the builds and make sure there are no bugs remaining.

**Deploy to the production phase**

Throughout the pipeline, whenever there is an error, feedback will be instantly sent to the development team so that issues are immediately addressed. Code changes to fix bugs will then go through the production pipeline once again. After codes or the product passed all the tests without defects, they move on to the production server in the final phase. The constant feedback loop helps make the pipeline a closed process where builds are continuously committed, tested, and deployed to production

**10 a) You want to have 2 version of your application in production, but be able to switch all the traffic between them. Explain the deployment strategy for the give situation with neat diagram. 10 Marks**

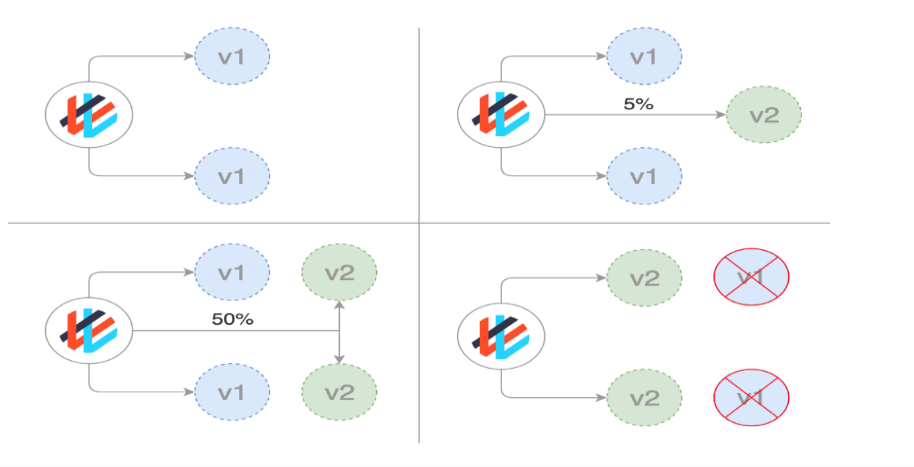
One possible deployment strategy for this situation is called **canary deployment.** In canary deployment, the deployment team sets up the new version and then gradually shifts the production traffic from the older version to the newer version. For example, at a point in time during the deployment process, the older version might retain 90% of all traffic for the software while the newer version hosts 10% of the traffic. This deployment technique helps the DevOps engineers test the stability of the new version.



Canary deployment enables better performance monitoring. It also aids in a faster and better rollback of the software if the new version fails. However, it has a slow nature and a more time-consuming deployment cycle.

A canary is used for when you want to test some new functionality typically on the backend of your application. Traditionally you may have had two almost identical servers: one that goes to all users and another with the new features that gets rolled out to a subset of users and then compared. When no errors are reported, the new version can gradually roll out to the rest of the infrastructure.

While this deploy strategy can be done just using Kubernetes resources by replacing old and new pods, it is much more convenient and easier to implement this strategy.



1. **b) Analyze and explain how Kubernetes works in automating a software deployment process. 10 Marks**

Kubernetes is an open-source Container Management platform which automates container deployment, container scaling, and descaling and load balancing Docker containers in the form of a cluster. Along with the automated deployment and scaling of containers, it provides healing by automatically restarting failed containers and rescheduling them when their hosts die. This capability improves the application’s availability.

**Steps how kubernetes automates software deployment process**

1. **Configure Deployment**

Create a Deployment manifest file with

* Container image(s)
* Number of replicas
* Resource requirements
* Health checks
* Other configuration options

1. **Create Deployment:**

\* Submit the manifest to Kubernetes API server (e.g., using `kubectl`).

\* Kubernetes control plane creates a ReplicaSet to manage Pods.

1. **Pod Scheduling:**

Scheduler selects suitable nodes to run Pods based on resource availability and constraints.

1. **Pod Creation & Management:**

\* Kubelet agent on each node creates and starts Pods.

\* Deployment controller monitors their health and status.

1. **Service Exposure:**

\* Create a Service to expose Pods to the network.

\* Service load balances traffic across multiple Pods for high availability.

1. **Updates & Rollouts:**

\* Modify the manifest with new image/configuration and submit.

\* Kubernetes creates a new ReplicaSet with updated configuration.

\* Deployment controller gradually scales down old and scales up new ReplicaSet for smooth transition.

\* Service automatically directs traffic to new Pods.

1. **Rollbacks:**

* In case of update failure, Kubernetes can automatically roll back to a previous stable version.

**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.



RASHMI T N

Lecturer, CSE

GPT Channapatna