V Semester Diploma Examination, September - 2023

Full Stack Development-20CS52I

Question Paper

Instructions: i) Answer one full question from each section.

ii) Each section carries 20 Marks.

SECTION-I

1. **a)** Define Enterprise. Explain the steps involved in the organizing an enterprise. -10

b) DigiLocker is a digital service app under Digital India Initiative. Explain how digital transformation has changed the way we handle our documents. -10

2. **a)** Explain the process of design thinking. Apply the process of design thinking that led to the evolution of smart phones. -10

b) Differentiate between 3 different cloud service models. -10

SECTION-II

3. **a)** IRCTC – A Train ticket booking application that helps the users to book a train ticket for travelling across India. This application allows users to login for booking tickets. Users can search for train between source and destination places for a particular day. Once found, user can check the availability of tickets for different classes like general, sleeper, AC etc. Users can book a particular train ticket on required date. Once a ticket is booked after making required amount through online payment mode, user can get the booking details. Identify and write the user stories for this application. -12

b) Write test cases for the above application. -8

4. **a)** GPay is a digital payment application that helps its users to do variety of money transaction. This application allows users to log in using their mobile and OTP for login validation. Once app is registered with the user bank account, users can do transaction either by scanning QR code, by contact mobile number, bank transfer or bill payment. Every transaction has to be authorized by 4-digit UPI pin. Once transaction is successful, user gets transaction details. User can check transaction history, bank balance, rewards and offers. Identify and write the user stories for this application. -12

b) Write test cases for the above application. -8

SECTION-III

5. **a)** Create Student.java – an entity class with student information. Design a Rest Controller with GET, PUT, POST and DELETE Restful APIs. -10

b) Describe react components. With a simple code explain different react components. 10

6. **a)** Perform the following operations using MongoDB

i. Create an employee database.
ii. Insert 2 records into the employee collection with suitable fields.
iii. Sort the employee by name
iv. Search the employee by Id
v. Remove the employee by Id -10

b) Explain state and props with an example in React JS. -10

SECTION-IV

7. **a)** Differentiate spring and spring boot. -10

b) Describe different spring annotations along with their usage and syntax -10

8. **a)** Implement programmatically navigation between different components using react Router. -10

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

SECTION-V

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

b) With a neat diagram explain CI/CD build process flow. -10

10. **a)** Discuss the Components of Docker Container. -10

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

Scheme of Valuation

Section-I

1. a) Definition -1 Mark, Steps List -2 Marks, Explanation -7 Marks

1. b) Digilocker Explanation -3 Marks. Explanation of Digital Transformation -7 Marks

2. a) 5 steps of Design Thinking Process- 5 marks, Smartphone Design Thinking - 5 Marks

2. b) 10 differences, each difference carries 1 mark -- 10*1=10 Marks.

Section-II

3. a) Identification of 4 User Stories -2 Marks, Writing of 4 User Stories -- 4*2.5=10 Marks

3. b) Writing of any 8 Test Cases - 8 Marks

4. a) Identification of 4 User Stories- 2 Marks, Writing of 4 User Stories -- 4*2.5=10 Marks

4. b) Writing of any 8 Test Cases - 8 Marks

Section-III

5. a) Entity class -2 Marks, 4 controller carries 2 marks each -- 4*2=8 Marks

b) React Components -2 Marks, Function component -4 Marks, React components -4 Marks

6. a) Each MongoDb operation carries 2 marks -- 2*5=10 Marks

6. b) State Explanation -2Marks, example -3Marks, Props Explanation -2Marks, example -3 Marks

Section-IV

7. a) any 5 Comparison -- 5*2=10 Marks

7. b) Each attributes carries 1 mark -- 1*10=10 Marks

8. a) Implementation 10 Marks

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

Section-IV

9. a) Any 5 difference, each carries 2 marks -- 5*2=10 Marks

9. b) CI/CD diagram -2 Marks, Explanation -8 Marks

10. a) Docker Client -3 Marks, Docker Host-5 Marks, Docker Registries -2 Marks

10. b) Blue Green Deployment Explanation-3 Marks Diagram-2 Marks, Canary Deployment Explanation -3 Marks, Diagram -2 Marks

Model Answers

Section-I

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

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.

1. b) DigiLocker is a digital service app under Digital India Initiative. Explain how digital transformation has changed the way we handle our documents. 10

DigiLocker is a digitization service provided under its Digital India initiative. DigiLocker allows access to digital versions of various documents including driver's licenses, vehicle registration certificates and academic mark sheets etc. User can request necessary document from various department. It also provides 1 GB storage space to each account to upload scanned copies of legacy documents.

Digital transformation is the integration of digital technology into all areas of a business, fundamentally changing how you operate and deliver value to customers. Digitization is transforming the world, improving our lives by making things simple. The following are the various ways that digital transformation has brought changes to the we handle our documents.

- **Paperless Document**: It reduces the use of physical documents and paper, which saves time, money and environmental resources.
- **Reliable**: The digital platform is an ambitious program created and administered by government of India.
- **Authentic Document**: It enables easy verification of documents by government agencies and departments, which reduces the risk of fraud and forgery.
- **Legal Document**: The documents provided and shared through Digi Locker are at par with the corresponding physical certificates.
- **Safe & Secure**: It ensures the safety of documents by storing them in a secure cloud platform that can be accessed only by the user or authorized entities.
- **Simple & Convenient**: It provides a convenient and seamless customer experience by allowing users to access, share and e-sign their documents anytime and anywhere using their mobile devices.

2. **a) Explain the process of design thinking. Apply the process of design thinking that led to the evolution of smart phones. 10**

Design thinking enables companies to test the feasibility of the future product and its functionality at the initial stage. It allows them to keep end user needs in mind, clearly specify all requirements and translate all this into product features.

Design Thinking is a 5-step process to come up with meaningful ideas that solve real problems for a particular group of people.

Empathize
Define
Ideate
Prototype
Test

The five steps in design thinking are:

- **Empathize**: Understand the user's needs and perspective.
- **Define**: Define the problem that needs to be solved.
- **Ideate**: Generate new and innovative ideas.
- **Prototype**: Create a prototype of the product.
- **Test**: Test the product with actual users to get feedback.

Design thinking that led to development of Smartphone.

- Empathizing user's need: The traditional mobile users are restricted to text and voice communication and are dependent on gadgets like computer/laptop for accessing web application and resources.
- Problem Definition: The need of a single device that combine the functions of a mobile phone, a personal computer, a camera, high end processor and other features.
- Ideate: The idea of Smartphone – a device capable of working like mobile & computer with touch screen, bigger display and many more features is generated.
- Prototype: The idea is transformed into working model i.e., the prototype is created.
- Test: The rigorous testing is done on prototype model to ensure user's needs are met.

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

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

3. **a) IRCTC – A Train ticket booking application that helps the users to book a train ticket for travelling across India** **12**

- **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 train booking app.

  Acceptance Criteria:

  i. While signing up, enter Username, 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.

- **User Story: Login** As a registered/authorized user, I want to login into IRCTC application so that I can access variety of service provide 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 "bad credentials".

  iii. After successful log in, home page is displayed.

- **User Story: Search Train** As an authorized user, I want to search train so that I can book ticket for my planned trip.

  Acceptance Criteria:

  i. User has to enter valid from(source), to(destination) train stations, valid date.

ii. A valid search displays list of trains with schedule.
iii. User can sort, filter and modify the search results.
iv. User can view available tickets in each travel class like AC, Sleeper etc.

- **User Story: Book Train** As an authorized user, I want to book train so that I can make reservation for particular place, date and time.
  Acceptance Criteria:
  i. User has to choose available train & travel class for ticket booking.
  ii. Passenger's details like name, age, gender, berth preference need to be entered
  iii. Valid contact information, travel insurance option & payment mode are required.
  iv. After successful payment, user should get the booking details to registered mobile number and E-mail id.
- **User Stories: Logout** As a logged in user, I want to log out of IRCTC app so that I can prevent unauthorized access of my 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**. **8**

- **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 numbers.
  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 Train Search:**

i. Ensure that the train search page displays list of trains corresponding to entered source and destination station for the specific date.
ii. Verify whether user is able to apply filter, sort trains and modify existing search.
iii. Verify whether user is able to select the required train, check for availability of tickets

- **Test Cases for the Train Booking:**

i. Ensure whether user is able to add passenger details for selected train.
ii. Verify the upper limit for number of passengers booked per train.
iii. Verify that the system displays the total cost of the ticket purchase as per booking details, including any taxes and fees.
iv. Verify whether user is able to cancel the ticket or partially cancel the tickets or not.
v. 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, PhonePay and mobile wallets
iii. Ensure that the payment gateway sends a confirmation email or SMS to the user after the successful transaction.

4 a. GPay is a digital payment application that helps its users to do variety of money transaction. 12

- **User Story: Registration/Sign-up**: As a new user, I want to sign up for the GPay app so that I can access the app for digital payment.

  Acceptance Criteria:

i. Mobile number linked to bank account should be used for app registration on smart phone.
ii. Choose google account associated with user smart phone.
iii. Verify the registered mobile number with OTP.
iv. Addition security has to enabled either by setting password or 4-digit pin.

- **User Story: Bank Account Linking** As a registered user, I want to link my bank account so that I can do any digital payment transaction.

  Acceptance Criteria:

i. Must add preferred bank name registered with user mobile number.
ii. App must find the account number linked to mobile number for the preferred bank.

iii. User should set 4 or 6-digit pin for authorizing the digital payment transaction.

- **User Story: Digital Payment** As a genuine user, I want to perform digital payment so that can manage my finances or pay my bills on time.

  Acceptance Criteria:

i. User must scan QR code to transfer money – amount will be automatically displayed or user must enter the amount.
ii. User must enter mobile number, amount to pay to his friends/contact.
iii. For bank transfer, user must enter account number, IFSC code, account holder's name. After successful linking, user can pay by entering desired amount.
iv. For bill payment, bill needs to be liked to service providers. Once liked user gets outstanding amount to be paid.
v. User must get confirmation message, notification for all the above digital payment.

- **User Stories: Utilities** As a user, I want to check my transaction, balance, offers & reward so that I can plan my next transaction.

  Acceptance Criteria:

i. User must get bank balance after validating security pin.
ii. User must be able to check the transaction history, filter transaction by name, date, phone number etc.
iii. User must be able to check, apply the offers and rewards provided in the app.

4. b) Write test cases for the above application. 8

- **Test Cases for the Registration Page**:

i. Verify whether user is able download Gpay app from play store and installation of app on different smartphone like android, ios etc is possible or not.
ii. Verify 4-digit pin is validated for login authentication.
iii. Check whether mobile number is verified with one time password.

- **Test Cases for the Bank Account Registration**

i. Check when mobile number is entered, bank details linked to registered mobile number is fetched or not.
ii. Check whether user is able to set 4-digit transaction pin for addition security.

- **Test Cases for the Digital Payment**

i. Ensure camera is opened for scanning when scan QR code option is selected.
ii. Ensure when valid mobile number is entered, authorized gpay recipient name is displayed.
iii. Check whether account number and IFSC validation are done for bank transfer.

iv. Check whether user is able to access list of bill provider for each category & verify his account linked to the same or not.

v. Ensure user gets acknowledgement for completion of every successful transaction.

- **Test Cases for the Utilities**

i. Check whether user is able to access/apply offer and rewards available on the app.

ii. Check whether user is able to retrieve bank balance from bank or not.

5. a) Create Student.java – an entity class with student information. Design a Rest Controller with GET, PUT, POST and DELETE Restful APIs. -10

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() {                           // Getters & Setters methods
                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 fees; }
        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 String id){
return studentservice.getStudentDetails(id);}
@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 String id){
studentservice.updateStudentDetails(student,id);}
@DeleteMapping("/students/")                                  //Delete Student Details
public List<Student> deleteStudentDetails(@RequestBody Student student){
studentservice.deleteStudentDetails(student);}
}
```

Note: Code for StudentService class is optional.

5. b) Describe react components. With a simple code explain different react components.

A component is considered as the core building blocks of a react application. It makes the task of building UIs much easier. React components are independent and reusable code. Components are similar to javascript functions, but work individually to return JSX code as elements for our UI.

- **Functional Components**

  ReactJS functional components are simple javascript functions starting with uppercase letter. These functions may or may not receive data as parameters. In the functional components, the return value is the JSX code to render to the DOM tree. It is known as Stateless components as they simply accept data and display them in some form, they are mainly responsible for rendering UI.

```
import React from 'react';
import ReactDOM from 'react-dom/client';
function Car() {                                              //Functional Component
return <h2>Hi, I am a Car!</h2>; }
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<Car />);
```

- **Class Components**

A class component must include the extends React. Component statement. The component also requires a render() method, this method returns HTML. It is also known

as stateful components because they implement logic and state.

```
import React from 'react';
import ReactDOM from 'react-dom/client';
class Car extends React.Component {                           //Inheritance
render() {                                                    //Render function
return <h2>Hi, I am a Car!</h2>;  }}
```

```
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<Car />);
```

6. a) Perform the following operations using MongoDB **10**

i. Create an employee 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 employee                    // DB Creation
show dbs                        // To display DB
```

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

The two 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().

```
db.employee.insertMany([{name:”abc”, emp_id:123,emp_dep:”mno”},
                        {name:”xyz”, emp_id:987,emp_dep:”pqr”}]
```

iii. Sort the employee 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.employee.find.sort({name:1})
```

iv. Search the employee by Id

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

```
db.employee.find({id:1})
```

v. Remove the employee by Id

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

```
db.epmloyee.deleteOne({id:1})
```

6. b) Explain state and props with an example in React JS. **10**

- State in ReactJS

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);
```

7. a) Differentiate spring and spring boot. -10

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

7. b) 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.

Note: Syntax is same as heading. Refer 5 a) program for more details.

8. a) Implement programmatical navigation between different components using reactRoute. 10

- 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;
```

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

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
```

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.

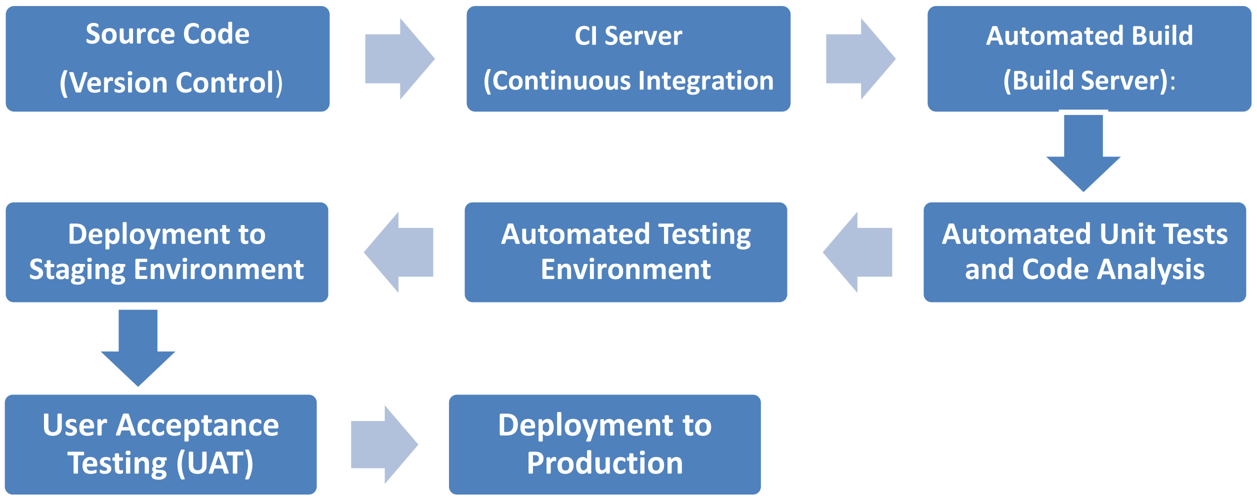
9. b) With a neat diagram explain CI/CD build process flow. -10

The CI/CD build process flow diagram for an online application typically involves multiple stages and components to automate the building, testing, and deployment of the application. Below is a simplified representation of the CI/CD process flow.

- **Source Code (Version Control)**: This is the central repository where developers store and manage the application's source code. Popular version control systems include Git, SVN, etc. Developers push code changes to this repository.
- **CI Server (Continuous Integration)**: The CI server monitors the version control system for code changes. Whenever a new commit is pushed or a pull request is submitted, the CI server is triggered. Its primary purpose is to automate the integration of code changes into a shared repository and perform various automated tasks.
- **Automated Build (Build Server)**: Upon triggering, the CI server initiates an automated build process. It compiles the source code, gathers dependencies, and generates a build artifact (e.g., executable, binary, or container image). This artifact represents the built application.
- **Automated Unit Tests and Code Analysis**: After the build, the CI server runs automated unit tests to check the functionality and correctness of the application. Additionally, it may perform static code analysis to identify potential issues, bugs, or code style violations.
- **Automated Testing Environment**: This is an isolated environment where the application is deployed for automated testing. It simulates the production environment but may have fewer resources. Automated integration tests, regression tests, and other tests are conducted here.
- **Deployment to Staging Environment**: If all the previous stages (build and automated tests) are successful, the application is deployed to a staging environment. The staging

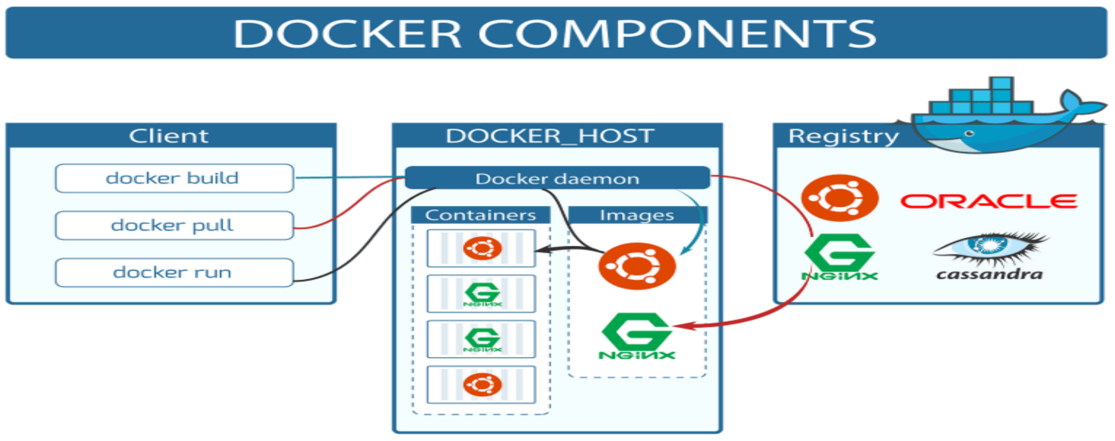
environment is a near-production replica where final testing is conducted before going live.

- **User Acceptance Testing (UAT)**: In this phase, the application is tested by actual users (typically non-technical stakeholders) to ensure it meets business requirements and user expectations.
- **Deployment to Production**: Upon successful UAT and approval, the application is deployed to the production environment, making it available to end-users.


Source Code
(Version Control)
CI Server
(Continuous Integration
Automated Build
(Build Server):
Automated Unit Tests and Code Analysis
Automated Testing Environment
Deployment to Staging Environment
User Acceptance Testing (UAT)
Deployment to Production

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

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 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.

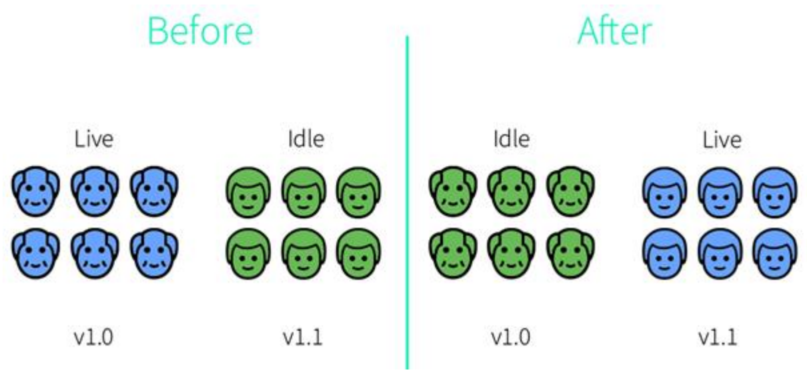
10. b) Illustrate the working Blue-Green and Canary deployment strategies with neat diagram. **10**

- **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.

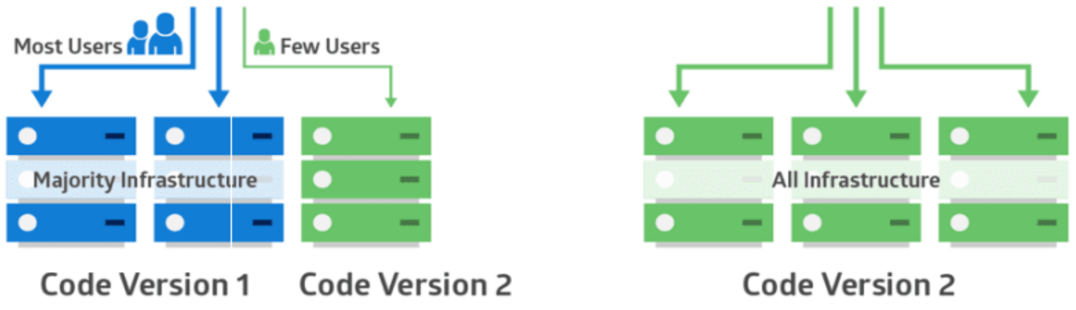


- **Canary deployment**

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.

[illegible]

GAGANDEEP M. N.
Lecturer in CS& E
Government Polytechnic, Bantwal