V Semester Diploma Examination, January/February- 2023

Programme: Computer Science and Engineering

Course: Full Stack Development-20CS52I

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

ii) Each section carries **20** Marks.

SECTION-I

1. a) Digital transformation is creating new or modifying existing processes. Explain how digital transformation can bring revolution in teaching learning process. **-10**

b) Digital payments are transactions like Gpay, Phonepe etc. that take place via digital or online modes, with no physical exchange of money involved. Explain how design thinking has brought revolution in digital payments. **-10**

2. a) Discuss the different steps involved in creating a project plan in an organization. **-10**

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

SECTION-II

3. a) Hotel Booking is an online Hotel room booking application that helps the users to book a room for staying at particular place across Karnataka. This application allows users to login for booking a room. Users can search for the room at a hotel for a specific location. Once found, user can check the availability of a room for specific dates. Users can book a hotel required date. Once booked, 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) Flipkart is an online shopping application that helps its users to buy variety of authentic products. This application allows users to log in for buying products. Users can search for a product, sort the product list based on rating or price. Users can select the items and add them to the cart. Once the selection is done, users can go to the cart page for payment. Identify and write the user stories for this application. **-12**

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

SECTION-III

5. a) Assume Customer.java file has already been created with code. Design:

a) Customer.dto file and **-10**

b) Customer.dao file **-10**

for inserting and deleting customer details.

6. a) CRUD operations describe the conventions of a user-interface that let users view, search, and modify parts of the database. Explain how the Create operation is used to insert new documents in the MongoDB database. **-10**

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

SECTION-IV

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

b) Illustrate how react class components and functional component play major role in front-end UI design. **-10**

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

b) Discuss the rules to follow for an API to the RESTful. **-10**

SECTION-V

9. a) Discuss Components of application performance management(APM). **-10**

b) Demonstrate CRUD operations in MongoDB. **-10**

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

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

Scheme of Valuation

Section-I

1. a) Explanation of Digital Transformation **10 Marks**

1. b) Explanation of Design Thinking **10 Marks**

2. a) Explanation of 9 steps-Each **1Marks=9*1=9Marks** and Diagram-**1 Marks**

2. b) Identification-**1 Marks** Characteristics and advantages-**1+1=2 Marks.**

Cisco WebEx- **1+1+1=3 Marks.** Google App Engine-**1+1+1=3 Marks**

Amazon EC2-**1+2+1=4 Marks**

Section-II

3. a) Identification of User Stories-**3 Marks** Writing of User Stories-**9 Marks**

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

4. a) Identification of User Stories-**3 Marks** Writing of User Stories-**9 Marks**

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

Section-III

5. a) Customer.dto-Inserting customer Details- **-10 Marks**

b) Customer.dao Removing Customer Details-**3 Marks** and Customer.daoImpl-**7 Marks**

6. a) Explanation of 2 insert operations- **2*5=10 Marks**

6. b) State Explanation-**3Marks** Example-**2 Marks**

Props Explanation-**3Marks** Example-**2 Marks**

Section-IV

7. a) Comparison- **10 Marks**

7. b) React Class Components-**5 Marks** Functional Components-**5 Marks**

8. a) Implementation **10 Marks**

8. b) List rules-**1 Marks** Explanation of 6 rules-**6*1.5=9 Marks**

Section-IV

9. a) Five Elements Explanation-Each 2 Marks. **5*2=10 Marks**

9. b) Explanation of 4 Crud operations- **2.5*4=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**

Section-I

1. a) Digital transformation is creating new or modifying existing processes. Explain how digital transformation can bring revolution in teaching learning process. -10 Marks

Digital transformation is the process of revolutionizing any processes through the adoption of digital technologies. Digital transformation in teaching is a philosophical and physical change created to meet the campus, faculty, and students' ever-increasing demands to create a learning environment where everything connects.

1. Improved accessibility and access

The most inspiring digital transformation is visible in education is the improved accessibility to school, lessons and even degree programs for students of all ages. Students who may have impairments that hinder their ability can access certain types of information through the available technology.

Online learning opportunities, which allow students to access schools and degree programs through online help students secure the best possible education for their given situation.

2. Personalized learning approaches

In Personalized learning approaches students are allowed to learn in a way that fits their own learning approaches best, it helps them absorb and retain critical information; personalization empowers them to move forward in their education.

When students feel this engaged with the material at hand, it also helps to keep them on track with their program. Schools/Colleges that can genuinely meet the needs of their students and keep them moving forward with the coursework will also improve their retention rates.

3. Virtual reality

In the world of education, virtual reality can provide students with the chance to 'experience' the material they learn before they actually move into real-world applications.

Example: In the hospitality industry, students can see firsthand the different potential work environments, feel as though they are in a situation where they need to serve customers, and receive training that gives them hands-on experience without even leaving their classroom. This can help students feel more comfortable and better prepared for their future careers and the completion of their education.

4. Cloud-based learning opportunities

The cloud offers students and teachers the chance to connect from virtually anywhere. They can use these types of applications while sitting in a lecture hall in person, from home, or even

half-way around the world.

Teachers enable the streaming of lectures, thus making online classes possible and interactive. Students can also use many applications to submit their assignments, track their syllabus, and even connect and engage with others in their class. The platforms can be used to break out into smaller groups so that students can collaborate together on projects and assignments.

5. Incorporating the Internet of Things into the College environment

IoT can benefit education campuses by helping colleges improve security and comfort features while controlling costs. Smart devices that allow college officials to understand traffic patterns will inform them where lights and security features will have the most value.

IoT can also help colleges remain connected with students. Time-stamps help track assignments so that students can better monitor their degree progress and verify their work has been received. Student tracking also helps teachers and professors take attendance and know when someone is missing.

6. Security across digital devices

Security has become another necessary trend in the digital education revolution. Colleges collecting a wealth of information on students, from their personal data to grades, and that information should be protected and secured.

Security protocols that allow Colleges to record, store and transmit sensitive student data will be critical throughout the digital transformation. Colleges will also want to make sure that they have a means of securely allowing students to digitally submit assignments and verify user authenticity.

7. Teaching digital citizenship

Digital transformation will impact students in order to know how to interact politely and civilly online. i.e.. students learning how to be professional in an office or hospital setting.

When educating students on being a good digital citizen, Colleges empower students to embrace the full capacity of technology. Institutions should aim to make digital citizenship as a part of the culture for the students and teachers. These principles guide people to behave more collaboratively online, which can help students succeed in their classes and in a professional environment.

8. Big data

Colleges have long collected a wealth of information about their students, including their demographics, grades and classes. Big data provides them with the opportunity to take this

information much further and use it to better understand student trends and successes.

Big data refers to the growing technological capabilities to track large amounts of data and interpret it with the assistance of algorithms to find patterns and helpful information.

Big data can track greater trends in student populations, including student performance and professional outcomes, how well students respond to different types of information delivery and class styles. The information collected will empower Colleges to better serve their students.

1. **b) Digital payments are transactions like Gpay, Phonepe etc. that take place via digital or online modes, with no physical exchange of money involved. Explain how design thinking has brought revolution in digital payments.**

Design thinking is a creative problem-solving approach that focuses on the needs of the customer. In banking, design thinking can be used to create products, services, and experiences that are more user-friendly and effective.

Digital payment is a way of payment which is made through digital modes. In digital payments, payer and payee both use digital modes to send and receive money. It is also called electronic payment. No hard cash (currency notes) is involved in the digital payments. All the transactions in digital payments are completed through online.

Design thinking in banking used to architect digital financial products in 5 steps:

Empathize, Define, Ideate, Prototype, Test.

Currently available digital payment systems include Banking cards, Digital wallets, Unified Payment Interface (UPI), Unstructured Supplementary Service Data (USSD), Immediate Payment Service (IMPS), Real Time Gross Settlement (RTGS), National Electronic Fund Transfer (NEFT), Aadhar Enabled Payment System (AEPS) and Mobile banking.

Open banking

1. Payment Cards- The most common types of payment cards are credit cards and debit cards. A payment card is electronically linked to an account or accounts belonging to the cardholder. These accounts may be deposit accounts or loan or credit accounts, and the card is a means of authenticating the cardholder.

The information required for using payment cards are Card Verification Value (CVV Number) and Expiry date of the payment card. The Payment cards are

Credit card: Central Bank of India was the first public bank to introduce Credit card. The issuer of a credit card creates a line of credit for the cardholder on which the cardholder can borrow. The cardholder can choose either to repay the full outstanding balance by the payment due date or to repay a smaller amount, not less than the "minimum amount", by that date.

Debit card: Debit card was introduced by Citi Bank .With a debit card, when a cardholder makes a purchase, funds are withdrawn directly from the cardholder's bank account.

Smartcard: Banks are adding chips to their current magnetic stripe cards to enhance security and offer new service, called Smart Cards. Smart Cards allow thousands of times of information storable on magnetic stripe cards.

Charge card: With charge cards, the cardholder is required to pay the full balance shown on the statement, which is usually issued monthly, by the payment due date. It is a form of short-term loan to cover the cardholder's purchases.

Fleet card: A fleet card is used as a payment card, most commonly for gasoline, diesel and other fuels at gas stations.

Gift card: A gift card also known as gift voucher or gift token is a prepaid stored-value money card usually issued by a retailer or bank to be used as an alternative to cash for purchases within a particular store or related businesses.

Store card: It is a credit card that is given out by a store and that can be used to buy goods at that store.

2. Unstructured Supplementary Service Data (USSD)-USSD is sometimes referred to as "Quick Codes" or "Feature codes". USSD is generally associated with real-time or instant messaging services. The user sends a request to the network via USSD, and the network replies with an acknowledgement of receipt.

The Information required for USSD transaction is MPIN/ IFSC/Aadhar number/Account number. Mobile Banking Personal Identification Number (MPIN) works as a password when we perform any transaction using Mobile.

3. Aadhaar Enabled Payment Service (AEPS)-The AEPS system leverages Aadhaar online authentication and enables Aadhaar Enabled Bank Accounts (AEBA) to be operated in anytime-anywhere banking mode through Micro ATMs. This system is controlled by the National Payments Corporation of India (NPCI).

For AEPS transaction information needed are.

1. Aadhaar Number **2.** Bank Issuer Identification Number (IIN) or Name

3. Finger Print

4. Unified Payments Interface (UPI)-Unified Payment Interface (UPI) is a new payment interface introduced by National Payments Corporation of India (NPCI).

Unified Payments Interface (UPI) is a system that powers multiple bank accounts to use several banking services like fund transfer, and merchant payments in a single mobile application.

A user can simply add all the bank accounts in a single UPI payment app without remembering or even typing banking user ID/Passwords. Each Bank provides its own UPI App for Android, Windows and iOS mobile platform(s). The information required for UPI based transaction are Virtual Payment Address (VPA) of recipient and Mobile banking Personal Identification Number (MPIN).

5. Digital Wallets- A Digital wallet is a way to carry cash in digital format. Credit card or debit card information should be linked to digital wallet application or money can be transferred in online to mobile wallet.

The Services offered by Digital Wallets are Balance Enquiry, Passbook/ Transaction history, Add money, Accept Money, Pay money etc. Digital wallets are composed of both digital wallet devices and digital wallet systems.

Ex: Paytm, Freecharge, Mobikwik, Oxigen, mRuppee, Airtel Money, Jio Money, SBI Buddy, itz Cash, Citrus Pay, Vodafone M-Pesa, Axis Bank Lime, ICICI Pockets, SpeedPay etc.

6. Mobile Banking- Mobile banking is a service provided by a bank or other financial institution that allows its customers to conduct different types of financial transactions remotely using a mobile device such as a mobile phone or tablet.

It uses software, usually called an app, provided by the banks or financial institution for the purpose. Each Bank provides its own mobile banking App for Android, Windows and iOS mobile platform(s).

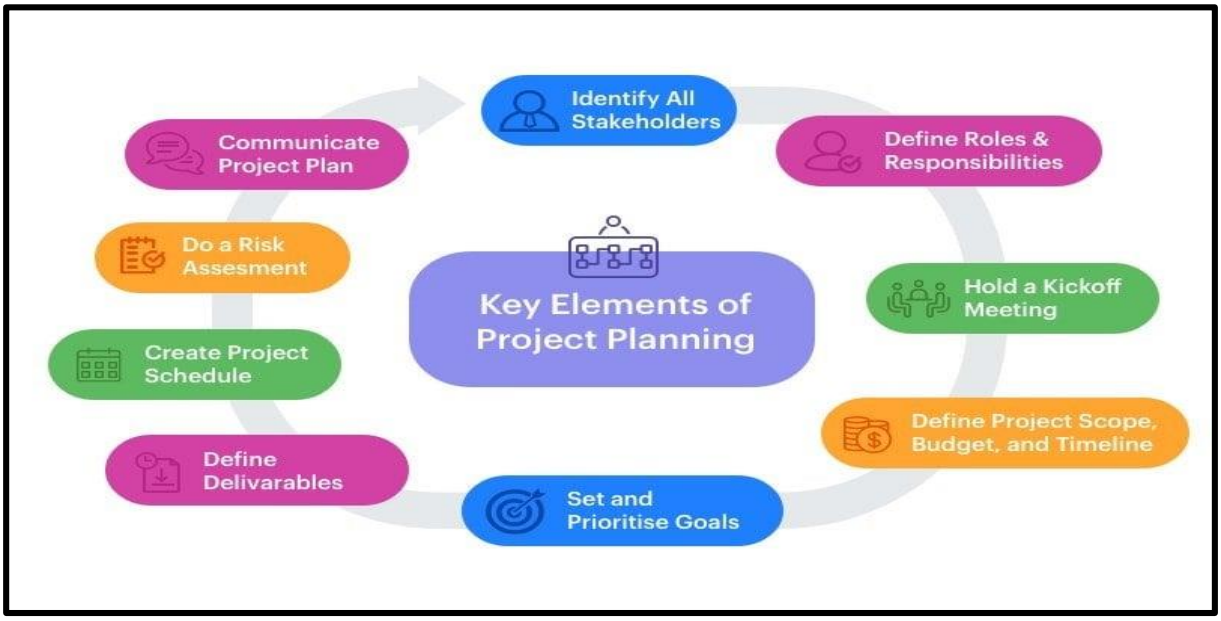
Ex: iMobile for ICICI bank, Kotak Bank App for Kotak Mahindra bank, SBI freedom app for State bank of India

7. Internet Banking- Internet banking, also known as online banking, e-banking or virtual banking, is an electronic payment system that enables customers of a bank or other financial institution to conduct a range of financial transactions through the financial institution's website. Online banking was first introduced in the early 1980s in New York, United States. Following are the services provided by Internet banking.

- Bill payment service
- Railway pass
- Recharging the prepaid phone
- Shopping
- Fund transfer through NEFT, IMPS, RTGS or ECS

2. a) Discuss the different steps involved in creating a project plan in an organization. -10

Project planning is a crucial stage that comes right after initiation in project management phases. Project constraints such as time, scope, and costs are discussed in the project planning process, and mitigation plans are developed after the identification of potential risks.



Step 1: Identify all stakeholders

Our project has several stakeholders, and not all of them will be involved in every detail of the project. Project stakeholders include

- Our customer,
- The end-users of the product,
- The company and its leaders, and
- The team working directly on the project.

Depending on the nature of the project, stakeholders may also include outside organizations or individual community members that will be affected by the project.

Step 2: Define roles and responsibilities

Once we have identified our stakeholders, we need to determine the core project management skills and competencies required for the project. When we have that list, we can define roles and assign responsibilities to individual stakeholders.

Typical roles include

Project sponsor,

Project manager,

and Project team members.

The different project team member roles will vary depending on our project, but be sure to include a vendor relations role and a customer relations role.

Step 3: Hold a kickoff meeting

The kickoff meeting is an effective way to bring stakeholders together to discuss the project. It is an effective way to initiate the planning process. It can be used to start building trust among the team members and ensure that everyone's idea are taken into account. Kickoff meetings also demonstrate commitment from the sponsor for the project.

Some of the topics that will be included in a kickoff meeting are:

- Business vision and strategy (from sponsor)
- Project vision (from sponsor)
- Roles and responsibilities
- Team building
- Team commitments
- How team makes decisions
- Ground rules

Step 4: Define project scope, budget, and timeline

After the official kickoff, we have to define three important concepts:

1. The project scope, **2.** Budget, and **3.**Timeline of our project.

Scope: Project scope tells us what are we going to do (and not do)?

Budget: It defines what the expected financial cost of the project is?

Timeline: The project timeline itemizes the phases of our project and the length of time we can reasonably expect them to be completed.

Step 5: Set and prioritize goals

Once our team understands the objectives of the project and we've identified the phases to meeting those objectives,

- Break down the big picture objectives of our project into individual goals and tasks,
- Prioritize tasks according to importance and dependencies, and
- Put a system in place to ensure corrective actions when goals aren't met on time.

Step 6: Define deliverables

A deliverable is "any unique and verifiable product, result, or capability to perform a service that is produced to complete a process, phase, or project".i.e.,, a deliverable could be, a product, result, or capability.

Project deliverables are determined by the project objectives and are an essential part of the project plan.

Step 7: Create a project schedule

A project schedule is a document that details

- The project timeline,
- The organizational resources required to complete each task, and
- Any other information critical to the team management.

To create a project schedule,

- Further, divide the phases of our project into individual tasks and activities,
- Determine dependencies,
- Sequence the activities, and
- Estimate the required resources and duration of each task.

Step 8: Do a risk assessment

A risk is a problem that may or may not arise over the course of our project. It's important to identify risks in project management and mitigate them at the project planning phase rather than be caught off guard later.

Areas of risk include:

a) Project Scope
b) Resources (personnel, financial, and physical)
c) Project delays and
d) Failures of Technology or Communication

Step 9: Communicate the project plan

Once we've compiled our project plan, it should communicate it clearly to the team and all other stakeholders. We may have created a project communication plan when we put together our project schedule. Establishing solid communications channels and expectations for project communication is crucial. As a project manager, be sure to model the kind of communication you expect from all stakeholders.

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

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

2. **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

3. 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) HotelBooking is an online Hotel room booking application that helps the users to book a room for staying at particular place across Karnataka. This application allows users to log in for booking a room. Users can search for the room at a hotel for a specific location. Once found, user can check the availability of a room for specific dates. Users can book a hotel for required date. Once booked, user can get the booking details. Identify and write the user stories for this application.

Registration

1. Sign-up:

As an unauthorized user, I want to sign up for the HotelBooking application through a sign-up form, so that I can access to book a room.

Acceptance Criteria:

1. While signing up-Use Name, Username, Email, and Password and Confirm Password.
2. If sign up is successful, it will get automatically logged in.

3. If I sign up with an incorrect detail which are specified in step1, I will receive an error message for incorrect information.
4. If we are trying to sign up with an existing email address, we will receive an error message saying "email exists."

2. Login

As an authorized user, I want to log in for HotelBooking application, so that I can have access to the application.

Acceptance Criteria:

1. While logging in, Username and password are required.
2. After successful log in, it will be redirected to the main page.
3. If we are trying to login with incorrect username or password, then error message will be displayed as "invalid login".

3. Searching a Room

As an authorized user, I want to search for a room in HotelBooking application, so that I can book a room in a specific location.

Acceptance Criteria:

1. While searching, Valid location should be specified.

2. Checking for a room at specific date always should be current date and ahead of the current date.

4. Booking Room

As an authorized user, I want to book a room in HotelBooking application, so that I can reserve the room in a specific location and date.

Acceptance Criteria:

1. While Booking, accommodation should be allotted according to the room size.

2. One should select the valid payment method based on the price of reserved room.

3. After successful payment one should get the booking details to registered mobile number and E-mail id.

5. Logout

As an authorized user, I want to log out of HotelBooking application, so that I can prevent unauthorized access of my profile.

Acceptance Criteria:

- When I log out of my account, I will be redirected to the log-in page.

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

1. User is able to access the Hotel Booking Home page.
2. Validate the hotel booking Home page is rendered correctly for **desktop** as per the design specifications.
3. Validate the hotel booking Home page is rendered correctly for **tablet** as per the design specifications.
4. Validate the hotel booking Home page is rendered correctly for a **mobile** device as per the design specifications.
5. Validate hotel search fields are visible on screen.
6. User searches for a holiday to any place across Karnataka for a family of 2 adults and 2 children and makes a payment (End to End Test).
7. User makes a successful payment for their hotel booking.
8. User makes an unsuccessful payment of their hotel booking.
9. Hotel Room Unavailability – User searches for dates that are unavailable and system recommends alternative dates or room types.
10. User wants to Amend an existing booking by adding an additional feature (e.g. increase length of stay / adding breakfast).
11. User wants to Verify the itinerary and print a Paper version.
12. User cancels their booking and system refunds money – Test Refund Conditions
13. User cancels their booking and system does NOT refund money – Test Refund Conditions
14. User wants to make a group booking
15. User wants to Validate Booking Page displays correct booking data– Visual check
16. Confirm Payment Page is displayed when user selects "Make Payment".
17. End to End Test of Hotel Booking Engine.

4. a) Flipkart is an online shopping application that helps its users to buy variety of authentic products. This application allows users to log in for buying products. Users can search for a product, sort the product list based on rating or price. Users can select the items and add them to the cart. Once the selection is done, users can go to the cart page for payment. Identify and write the user stories for this application.

Registration

Sign-up:

As a shopper, I want to sign up for the Flipkart application through a New user form, so that I can get access to buy a variety of products.

Acceptance Criteria:

1. While signing up-Valid Phone Number/Email Id and OTP/Password.

2. If sign up is successful, it will get automatically logged in.
3. If I am trying to sign up with an invalid phone number/Email Id, I will receive an error message to enter a valid information.
4. If we are trying to sign up with an existing phone number/Email Id, we will receive an error message saying "you are already registered."

Login

As an authorized shopper, I want to log in for Flipkart application, so that I can have access to the application for searching and buying products.

Acceptance Criteria:

1. While logging in, Phone number/Email Id and OTP/Password are required.
2. After successful log in, it will be redirected to the main page.
3. If we are trying to login with incorrect mobile number/Email Id or OTP/Password, then error message will be displayed as "invalid credentials".

View a List of Products

As a Shopper I want to view a list of products so I can select some items to purchase.

Acceptance Criteria:

1. See a thumbnail image for each product
2. Click to view details for product
3. Add to cart from detail page
4. Search for a product
5. View products by category

Review a Cart

As a Shopper I want to review my cart so I can make adjustments prior to checkout

Acceptance Criteria:

1. View quantities and items in the cart
2. See a total cost before tax and shipping
3. Remove items
4. Adjust quantity of items
5. Click to navigate to a product detail page

Check out

As a Shopper I want to check out so I can get my products shipped to me.

Acceptance Criteria:

1. Trigger checkout from any page, if there are items in the cart
2. Enter a shipping address
3. Enter a billing address
4. Enter a credit card number
5. Show total including tax and shipping before finalizing.
6. Show Confirmation message after finalizing
7. Verify payment through our payment processor

Review Orders

As a Shopper I want to review my orders so I can see what I have purchased in the past.

Acceptance Criteria:

1. View a list of open and completed orders
2. See the status of the order
3. Navigate to the details of the order
4. Include a tracking number if the order is shipped but not delivered
5. Contact customer service about an order from the details page

Logout

As a Shopper, I want to log out of Flipkart application, so that I can prevent unauthorized access of my profile.

Acceptance Criteria:

When I log out of my account, I will be redirected to the log-in page.

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

Test Cases for Flipkart Website Login Page

- Check that there are proper validations on Login Page.
- Check for an error message if the Email, password, or any required field is left blank.
- Check the validations on the Email and password.
- When you log in, check that you stay logged in as you browse products. Also, you need to test the behavior.
- When the user doesn't interact with the site for some time, then the user has been logged out after the session times out.
- When you are logged in, log out and make sure you are logged out and that you cannot access any of the account's pages.

Test Cases for Flipkart Website Search Page

- Check that the products displayed are related to what was searched for.
- Check that the products should display an image, name, price, and maybe customer ratings and number of reviews.
- Check the more relevant product for the search term is displayed on the top for a particular search term.
- Check that all items on the next page are different from the previous page, i.e., no duplicates.
- Check that when both sorts and filters have been applied, they remain as we paginate or more products are loaded.
- Check that count of products is correctly displayed on the search result page for a particular search term.
- Check that filtering functionality correctly filters products based on the filter applied.
- Check that filtering works correctly on category pages.
- Check that filtering works correctly on the search result page.
- Check that the correct count of total products is displayed after a filter is applied.
- Check that all the sort options work correctly – correctly sort the products based on the sort option chosen.
- Check that sorting works correctly on the category pages.
- Check that sorting works correctly on the search result page.
- Check that sorting works correctly on the pages containing the filtered result, after applying filters.
- Check that the product count remains intact irrespective of the sorting option applied.

Test Cases for Flipkart Website Shopping Cart

- Users should be able to add a product to the cart.
- Item count should be incremented when the user adds the same product again.
- Taxes should be applied according to the delivery location.
- Users should be able to add items to the cart.
- Users should be able to update items in the cart.
- Checkout should happen successfully for the items added to the cart.
- Shipping costs for different products are added to the cart.
- Coupons should be applied successfully to the cart.
- The cart should retain the items even when the app is closed.

Payment Page Test Cases

- Check that After fill the shipping address and payment, the product is purchased successfully.
- Check that Different payment types should be present, e.g., Credit Card, PayPal, Bank Transfers, Installments, etc.
- Check the security of the client's card details when entered for payment.

Test Case for Flipkart Website – Post Order Page

- Email and order id should be sent after placement of order.
- Users should be able to cancel the order.
- There should be a facility for users to track the order.
- Users should be able to return/replace the product post-delivery.

Section-III

5. a) Assume Customer.java file has already been created with code. Design:

a) Customer.dto file and -10

b) Customer.dao file -10

for inserting and deleting customer details.

For the given question I have assumed the details for customer as Name, Phone Number, Age, Gender, Address and planid with some telephone company by name **infytel.**

1. Assume a file was created as Customer.java with some code which accepts all basic details.

2. Create a Customer.DTO file which includes all details and accept customer entity.

3. Create a file Customer.Dao java which calls method to insert and remove customer.

4. Create a file CustomerDAOImpl.java and connect to a database.

Write the code in "CustomerDTO.java" as below:

```
package com.infytel.dto;
import com.infytel.entity.Customer;
public class CustomerDTO {
    private Long phoneNumber;
    private String name;
    private Integer age;
    private Character gender;
```

```
    private String address;
    private Integer planId;
    public CustomerDTO() {}
    public CustomerDTO(Long phoneNumber, String name, Integer age, Character gender,
String address, Integer planId) {
        super();
        this.phoneNumber = phoneNumber;
        this.name = name;
        this.age = age;
        this.gender = gender;
        this.address = address;
        this.planId = planId;
    }
    public Long getPhoneNumber() {
        return phoneNumber;
    }
    public void setPhoneNumber(Long phoneNumber) {
        this.phoneNumber = phoneNumber;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public Integer getAge() {
        return age;
    }
    public void setAge(Integer age) {
        this.age = age;
    }
    public Character getGender() {
        return gender;
    }
    public void setGender(Character gender) {
        this.gender = gender;
```

```
	}
	public String getAddress() {
		return address;
	}
	public void setAddress(String address) {
		this.address = address;
	}
	public Integer getPlanId() {
		return planId;
	}
	public void setPlanId(Integer planId) {
		this.planId = planId;
	}
	@Override
	public String toString() {
		return "Customer [phoneNumber=" + phoneNumber + ", name=" + name + ",
age=" + age + ", gender=" + gender + ", address=" + address + ", planId=" + planId + "]";
	}

	public static Customer prepareCustomerEntity(CustomerDTO customerDTO)
	{
		Customer customerEntity = new Customer();
		customerEntity.setPhoneNumber(customerDTO.getPhoneNumber());
		customerEntity.setName(customerDTO.getName());
		customerEntity.setGender(customerDTO.getGender());
		customerEntity.setAge(customerDTO.getAge());
		customerEntity.setAddress(customerDTO.getAddress());
		customerEntity.setPlanId(customerDTO.getPlanId());
		return customerEntity;
	}
}
```

Write the code in an interface "CustomerDAO.java" as below:

```
package com.infytel.repository;
import com.infytel.entity.Customer;
public interface CustomerDAO {
```

```
    // Method to insert a Customer record into the db
    public void insert(Customer customer);
    // Method to remove a Customer record from the db
    public int remove(Long phoneNo);
}
```

Write the code in "CustomerDAOImpl.java" as below:

```
package com.infytel.repository;
import java.io.FileInputStream;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.Properties;
import org.apache.log4j.Logger;
import com.infytel.entity.Customer;
public class CustomerDAOImpl implements CustomerDAO {
    static Logger logger = Logger.getLogger(CustomerDAOImpl.class);
    Connection con = null;
    PreparedStatement stmt = null;
    ResultSet rs = null;
    @Override
    public void insert(Customer customer) {
        try (FileInputStream fis = new
FileInputStream("resources/application.properties");) {
            Properties p = new Properties();
            p.load(fis);
            String dname = (String) p.get("JDBC_DRIVER");
            String url = (String) p.get("JDBC_URL");
            String username = (String) p.get("USER");
            String password = (String) p.get("PASSWORD");
            Class.forName(dname);
            // Register driver
            con = DriverManager.getConnection(url, username, password);
            // Create connection
            String query = "insert into customer values (?,?,?,?,?,?)";
```

```
                    // Create prepared statement
                    stmt = con.prepareStatement(query);
                    stmt.setLong(1, customer.getPhoneNumber());
                    stmt.setString(2, customer.getName());
                    stmt.setInt(3, customer.getAge());
                    stmt.setString(4, customer.getGender().toString());
                    stmt.setString(5, customer.getAddress());
                    stmt.setInt(6, customer.getPlanId());
                    // Execute query
                    stmt.executeUpdate();
                    logger.debug("Record inserted");
            } catch (Exception e) {
                    logger.error(e.getMessage(), e);
            } finally {
                    try {
                            // Close the prepared statement
                            stmt.close();
                            // Close the connection
                            con.close();
                    } catch (Exception e) {
                            logger.error(e.getMessage(), e);
                    }
            }
    }
    @Override
    public int remove(Long phoneNo) {
            int result = 1;
            try (FileInputStream fis = new
FileInputStream("resources/application.properties");) {
                    Properties p = new Properties();
                    p.load(fis);
                    String dname = (String) p.get("JDBC_DRIVER");
                    String url = (String) p.get("JDBC_URL");
                    String username = (String) p.get("USER");
                    String password = (String) p.get("PASSWORD");
                    Class.forName(dname);
```

```
                con = DriverManager.getConnection(url, username, password);
                String query = "DELETE FROM Customer WHERE phone_no = ?";
                stmt = con.prepareStatement(query);
                stmt.setLong(1, phoneNo);
                result = stmt.executeUpdate();
            } catch (Exception e) {
                logger.error(e.getMessage(), e);
            } finally {
                try {
                    stmt.close();
                    con.close();
                } catch (Exception e) {
                    logger.error(e.getMessage(), e);
                }
            }
            return result;
        }
    }
```

6. a) CRUD operations describe the conventions of a user-interface that let users view, search, and modify parts of the database. Explain how the Create operation is used to insert new documents in the MongoDB database. -10

For MongoDB CRUD, if the specified collection doesn't exist, the create operation will create the collection when it's executed. Create operations in MongoDB target a single collection, not multiple collections. Insert operations in MongoDB are atomic on a single document level.

Create Operations

MongoDB provides two different create operations that you can use to insert documents into a collection:

1. db.collection.insertOne()
2. db.collection.insertMany()

insertOne()

insertOne() allows us to insert one document into the collection.

Example: We are considering a collection called RecordsDB. We can insert a single entry into our collection by calling the insertOne() method on RecordsDB. We then provide the

information we want to insert in the form of key-value pairs, establishing the schema.

```
db.RecordsDB.insertOne({
   name: "Mahesh",
   age: "8 years",
   species: "Dog",
   Address: "Kuvempunagar, Mysore",
   chipped: true
})
```

If the create operation is successful, a new document is created. The function will return an object where “acknowledged” is “true” and “insertID” is the newly created “ObjectId.”

```
> db.RecordsDB.insertOne({
... name: "Mahesh",
... age: "8 years",
... species: "Dog",
... Address: " Kuvempunagar, Mysore ",
... chipped: true
... })
{
      "acknowledged" : true,
      "insertedId" : ObjectId("5fd989674e6b9ceb8665c57d")
}
```

insertMany()

We can insert multiple items at one time by calling the insertMany() method on the desired collection. In this case, we pass multiple items into our chosen collection (RecordsDB) and separate them by commas. Within the parentheses, we use brackets to indicate that we are passing in a list of multiple entries. This is commonly referred to as a nested method.

```
db.RecordsDB.insertMany([{
   name: "Mahesh",
   age: "8 years",
   species: "Dog",
   Address: " Kuvempunagar, Mysore ",
```

```
  chipped: true},
   {name: "Keerthana",
   age: "12 years",
   species: "Cat",
   Address: "Jayangar, Bangalore”,
   chipped: true}])

db.RecordsDB.insertMany([{ name: "Mahesh", age: "8 years", species: "Dog",
Address: " Kuvempunagar, Mysore ", chipped: true},
{name: "Keerthana", age: "12 years",
species: "Cat", Address: " Jayangar, Bangalore ", chipped: true}])
{
      "acknowledged" : true,
      "insertedIds" : [
            ObjectId("5fd98ea9ce6e8850d88270b4"),
            ObjectId("5fd98ea9ce6e8850d88270b5")
      ]
}
```

6. b) Discuss 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 and these changes determine the behavior of the component and how it will render.

```
class Greetings extends React.Component {
 state = {
  name: "World"
 };
 updateName() {
  this.setState({ name: "Simplilearn" });
 }
 render() {
    return(
       <div>
```

```
            {this.state.name}
          </div>
      )
  }
}
```

- A state can be modified based on user action or network changes
- Every time the state of an object changes, React re-renders the component to the browser
- The state object is initialized in the constructor
- The state object can store multiple properties
- this.setState() is used to change the value of the state object
- setState() function performs a shallow merge between the new and the previous state

The setState() Method

State can be updated in response to event handlers, server responses, or prop changes. This is done using the setState() method.

The setState() method enqueues all of the updates made to the component state and instructs React to re-render the component and its children with the updated state.

Always use the setState() method to change the state object, since it will ensure that the component knows it's been updated and calls the render() method.

Example:

```
class Bike extends React.Component {
  constructor(props) {
    super(props);
     this.state = {
      make: "Yamaha",
      model: "R15",
      color: "blue"
    };
  }
  changeBikeColor = () => {
    this.setState({color: "black"});
  }
  render() {
    return (
      <div>
        <h2>My {this.state.make}</h2>
        <p>
```

```
        It is a {this.state.color}
        {this.state.model}.
      </p>
      <button
        type="button"
        onClick={this.changeBikeColor}
      >Change color</button>
    </div>
  );
 }
}
```

Props

React Props are like function arguments in JavaScript and attributes in HTML.

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. It allows passing data from one component to other components.

It is similar to function arguments and can be passed to the component the same way as arguments passed in a function. Props are immutable so we cannot modify the props from inside the component.

To send props into a component, use the same syntax as HTML attributes:

Example

Add a "brand" attribute to the Car element:

```
const myElement = <Car brand="Ford" />;
```

The component receives the argument as a props object:

Example

Use the brand attribute in the component:

```
function Car(props)
{
  return <h2>I am a { props.brand }!</h2>;
}
```

Pass Data

Props are also how you pass data from one component to another, as parameters.

Example

Send the "brand" property from the Garage component to the Car component:

```
function Car(props) {
  return <h2>I am a { props.brand }!</h2>;
}
function Garage() {
  return (
    <>
      <h1>Who lives in my garage?</h1>
      <Car brand="Ford" />
    </>
  );
}
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<Garage />);
```

Example

Create a variable named carName and send it to the Car component:

```
function Car(props) {
  return <h2>I am a { props.brand }!</h2>;
}

function Garage() {
  const carName = "Swift";
  return (
    <>
      <h1>Who lives in my garage?</h1>
      <Car brand={ carName } />
    </>
  );
}
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<Garage />);
```

Or if it was an object:

Example: Create an object named carInfo and send it to the Car component:

```
function Car(props) {
  return <h2>I am a { props.brand.model }!</h2>;
```

```
}
function Garage() {
  const carInfo = { name: "Swift", model: "Maruti Suzuki" };
  return (
    <>
      <h1>Who lives in my garage?</h1>
      <Car brand={ carInfo } />
    </>
  );
}
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<Garage />);
```

SECTION-IV

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

Spring	Spring Boot
Spring Framework is a widely used Java EE framework for building applications.	Spring Boot Framework is widely used to develop REST APIs.
It simplify Java EE development that makes developers more productive.	It shorten the code length and provide the easiest way to develop Web Applications.
The primary feature of the Spring Framework is dependency injection.	The primary feature of Spring Boot is Autoconfiguration. It automatically configures the classes based on the requirement.
It helps to develop loosely coupled applications.	It helps to create a stand-alone application with less configuration.
The developer writes a lot of code (boilerplate code) to do the minimal task.	It reduces boilerplate code.
To test the Spring project, we need to set up the sever explicitly.	Spring Boot offers embedded server such as Jetty and Tomcat, etc.
It does not provide support for an in-memory database.	It offers several plugins for working with an embedded and in-memory database such as H2.
Developers manually define dependencies for the Spring project in pom.xml.	Spring Boot use the concept of starter in pom.xml file that internally takes care of downloading the dependencies JARs based on Spring Boot Requirement.

b) Illustrate how react class components and functional component play major role in front-end UI design. -10

Components are independent and reusable bits of code. They serve the same purpose as **JavaScript** functions, but work in isolation and return **HTML**. They accept arbitrary inputs (called "props") and return **React** elements describing what should appear on the screen.

They are reusable and can be nested inside other components. So it creates the tree like structure, where our main component is connected with a small independent component and combining all we get the final output and in our default react app '**index.js**' is the main node where we import our '**app.js**'.

The component name must start with an Upper case letter.

For example, <Component>,<Example> etc. Cause, lower case are used for tags like ,<body>,<h1> etc.

In react we mainly have 2 types of components:

1. Function Components.
2. Class Components.

Functional Components

Functional components are more often used while working in **React**. These are simply **JavaScript** functions. We can create a functional component in **React** by writing a **JavaScript** function. In the functional Components, the return value is the **JSX** code to render(display) to the DOM tree.

Example:

1.

```
function Democomponent() {
  return <h2>This is function!</h2>;
}
export default Democomponent;
```

2.

```
const Democomponent=()=>
{
   return <h1>This is function!</h1>;
}
export default Democomponent;
```

Class Components

The class components are a little more complex than the functional components. The class components can work with each other. We can pass data from one class component to other class components. We can use **JavaScript ES6** classes to create class-based components in **React**.

Example:

```
import React from "react";
class App extends React.Component {
  render() {
    return <h1>This is component</h1>;
  }
}
export default App;
```

Functional Components Vs Class Components

- A functional component accepts props as an argument and returns a **React** element where, a class component requires us to extend from **React**. Component and create a render function that returns a **React** element.
- Functional components are stateless components as they simply accept data and display them in some form, that they are mainly responsible for rendering **UI** and Class components also known as Stateful components because they implement logic and state.

8. a) Implement programmatical navigation between different components using react Router. -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
```

b) Discuss the rules to follow for an API to the RESTful. -10

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

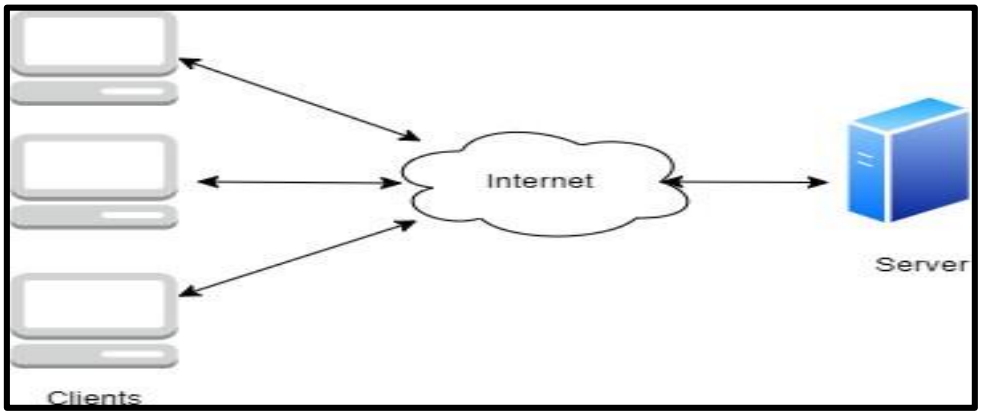
The API should facilitate communication between the client and server as they exchange data. To efficiently exchange data, we need a uniform interface.

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

2. 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.

Well-designed relationship shouldn't need to be updated every time applications on each end change.



3. 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.

4. 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 beenfully spent it will use the stored version.

5. 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.

6. Code on Demand

Code-on-Demand (COD) is the only optional constraint in REST. It allows clients to improveits flexibility because, in fact, it is the server who decides how certain things will be done. Forexample, 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 thisprocess.

SECTION-V

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

The five most important elements of application performance management are

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

1. End-User Experience Monitoring

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

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

2. Runtime Application Architecture Discovery

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

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

3. User-Defined Transaction Profiling

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

they traverse the architecture.

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

4. Component Deep-Dive Monitoring

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

5. Analytics

Finally, as with any IT scenario, having information is one thing; understanding it is another.

APM analytics tools allow federal IT pros to:

- Set a performance baseline that provides an understanding of current and historical performance, and set an expectation of what a "normal" application workload is
- Quickly identify, pinpoint, and eliminate application performance issues based on historical/baseline data.
- Anticipate and mitigate potential future issues through actionable patterns.
- Identify areas for improvement by mapping infrastructure changes to performance changes.

b) Demonstrate CRUD operations in MongoDB. -10

The basic methods of interacting with a MongoDB server are called CRUD operations. CRUD stands for Create, Read, Update, and Delete. CRUD operations describe the conventions of a user-interface that let users view, search, and modify parts of the database.

The individual CRUD operations:

- The Create operation is used to insert new documents in the MongoDB database.
- The Read operation is used to query a document in the database.
- The Update operation is used to modify existing documents in the database.
- The Delete operation is used to remove documents in the database.

Create Operations

The create or insert operations are used to insert or add new documents in the collection. If a collection does not exist, then it will create a new collection in the database. We can perform, create operations using the following methods provided by the MongoDB:

Method	Description
db.collection.insertOne()	It is used to insert a single document in the collection.
db.collection.insertMany()	It is used to insert multiple documents in the collection.
db.createCollection()	It is used to create an empty collection.

Example 1: In this example, we are inserting details of a single student in the form of document in the student collection using db.collection.insertOne() method.

```
> db.student.insertOne({
... name : "Sumit",
... age : 20,
... branch : "CSE",
... course : "C++ STL",
... mode : "online",
... paid : true,
... amount : 1499
... })
{
        "acknowledged" : true,
        "insertedId" : ObjectId("5e540cdc92e6dfa3fc48ddae")
}
>
```

Example 2: In this example, we are inserting details of the multiple students in the form of documents in the student collection using db.collection.insertMany() method.

```
> db.student.insertMany([
... {
... name : "Sumit",
... age : 20,
... branch : "CSE",
... course : "C++ STL",
... mode : "online",
... paid : true,
... amount : 1499
... },
...
... {
... name : "Rohit",
... age : 21,
... branch : "CSE",
... course : "C++ STL",
... mode : "online",
... paid : true,
... amount : 1499
... }
...
... ])
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("5e540d3192e6dfa3fc48ddaf"),
                ObjectId("5e540d3192e6dfa3fc48ddb0")
        ]
}
>
```

Read Operations

The Read operations are used to retrieve documents from the collection, or in other words, read operations are used to query a collection for a document. We can perform read operation using the following method provided by the MongoDB:

Method	Description
db.collection.find()	It is used to retrieve documents from the collection.

Example: In this example, we are retrieving the details of students from the student collection using db.collection.find() method.

```
> db.student.find().pretty()
{
        "_id" : ObjectId("5e540cdc92e6dfa3fc48ddae"),
        "name" : "Sumit",
        "age" : 20,
        "branch" : "CSE",
        "course" : "C++ STL",
        "mode" : "online",
        "paid" : true,
        "amount" : 1499
}
{
        "_id" : ObjectId("5e540d3192e6dfa3fc48ddaf"),
        "name" : "Sumit",
        "age" : 20,
        "branch" : "CSE",
        "course" : "C++ STL",
        "mode" : "online",
        "paid" : true,
        "amount" : 1499
}
{
        "_id" : ObjectId("5e540d3192e6dfa3fc48ddb0"),
        "name" : "Rohit",
        "age" : 21,
        "branch" : "CSE",
        "course" : "C++ STL",
        "mode" : "online",
        "paid" : true,
        "amount" : 1499
}
>
```

Update Operations

The update operations are used to update or modify the existing document in the collection. We can perform update operations using the following methods provided by the MongoDB:

Method	Description
db.collection.updateOne()	It is used to update a single document in the collection that satisfy the given criteria.
db.collection.updateMany()	It is used to update multiple documents in the collection that satisfy the given criteria.
db.collection.replaceOne()	It is used to replace single document in the collection that satisfy the given criteria.

Example 1: In this example, we are updating the age of Sumit in the student collection using db.collection.updateOne() method.

```
> db.student.updateOne({name: "Sumit"},{$set:{age: 24 }}
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCo
> db.student.find().pretty()
{
        "_id" : ObjectId("5e540cdc92e6dfa3fc48ddae"),
        "name" : "Sumit",
        "age" : 24,
        "branch" : "CSE",
        "course" : "C++ STL",
        "mode" : "online",
        "paid" : true,
        "amount" : 1499
}
{
        "_id" : ObjectId("5e540d3192e6dfa3fc48ddaf"),
        "name" : "Sumit",
        "age" : 20,
        "branch" : "CSE",
        "course" : "C++ STL",
        "mode" : "online",
        "paid" : true,
        "amount" : 1499
}
{
        "_id" : ObjectId("5e540d3192e6dfa3fc48ddb0"),
        "name" : "Rohit",
        "age" : 21,
        "branch" : "CSE",
        "course" : "C++ STL",
        "mode" : "online",
        "paid" : true,
        "amount" : 1499
}
>
```

Example 2: In this example, we are updating the year of course in all the documents in the student collection using db.collection.updateMany() method.

```
> db.student.updateMany({}, {$set: {year: 2020}})
{ "acknowledged" : true, "matchedCount" : 3, "modifiedCount" : 3 }
> db.student.find().pretty()
{
        "_id" : ObjectId("5e540cdc92e6dfa3fc48ddae"),
        "name" : "Sumit",
        "age" : 24,
        "branch" : "CSE",
        "course" : "C++ STL",
        "mode" : "online",
        "paid" : true,
        "amount" : 1499,
        "year" : 2020
}
{
        "_id" : ObjectId("5e540d3192e6dfa3fc48ddaf"),
        "name" : "Sumit",
        "age" : 20,
        "branch" : "CSE",
        "course" : "C++ STL",
        "mode" : "online",
        "paid" : true,
        "amount" : 1499,
        "year" : 2020
}
{
        "_id" : ObjectId("5e540d3192e6dfa3fc48ddb0"),
        "name" : "Rohit",
        "age" : 21,
        "branch" : "CSE",
        "course" : "C++ STL",
        "mode" : "online",
        "paid" : true,
        "amount" : 1499,
        "year" : 2020
}
>
```

Delete Operations

The delete operation are used to delete or remove the documents from a collection. We can perform delete operations using the following methods provided by the MongoDB:

Method	Description
db.collection.deleteOne()	It is used to delete a single document from the collection that satisfy the given criteria.
db.collection.deleteMany()	It is used to delete multiple documents from the collection that satisfy the given criteria.

Example 1: In this example, we are deleting a document from the student collection using db.collection.deleteOne() method.

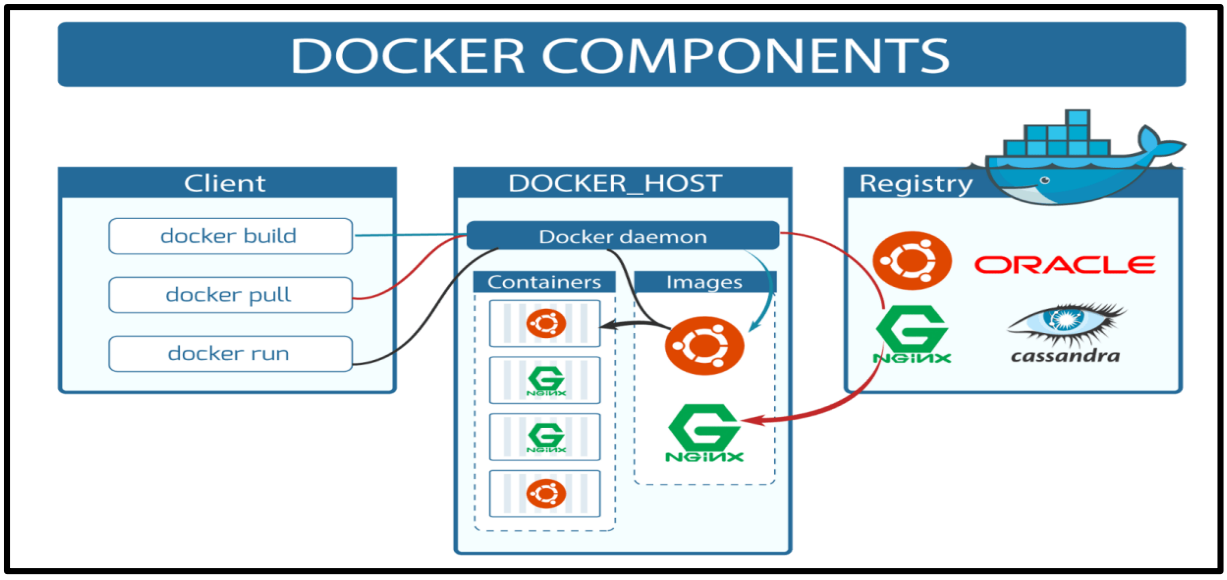
```
        "name" : "Sumit",
        "age" : 24,
        "branch" : "CSE",
        "course" : "C++ STL",
        "mode" : "online",
        "paid" : true,
        "amount" : 1499,
        "year" : 2020
}
{
        "_id" : ObjectId("5e540d3192e6dfa3fc48ddaf"),
        "name" : "Sumit",
        "age" : 20,
        "branch" : "CSE",
        "course" : "C++ STL",
        "mode" : "online",
        "paid" : true,
        "amount" : 1499,
        "year" : 2020
}
{
        "_id" : ObjectId("5e54103592e6dfa3fc48ddb1"),
        "name" : "Rohit",
        "age" : 21,
        "branch" : "CSE",
        "course" : "C++ STL",
        "mode" : "online",
        "paid" : true,
        "amount" : 1499
}
> db.student.deleteOne({name: "Sumit"})
{ "acknowledged" : true, "deletedCount" : 1 }
> db.student.find().pretty()
{
        "_id" : ObjectId("5e540d3192e6dfa3fc48ddaf"),
        "name" : "Sumit",
        "age" : 20,
        "branch" : "CSE",
        "course" : "C++ STL",
        "mode" : "online",
        "paid" : true,
        "amount" : 1499,
        "year" : 2020
}
{
        "_id" : ObjectId("5e54103592e6dfa3fc48ddb1"),
        "name" : "Rohit",
        "age" : 21,
        "branch" : "CSE",
        "course" : "C++ STL",
        "mode" : "online",
        "paid" : true,
        "amount" : 1499
}
```

Example 2: In this example, we are deleting all the documents from the student collection using db.collection.deleteMany() method.

```
> db.student.find().pretty()
{
        "_id" : ObjectId("5e540d3192e6dfa3fc48ddaf"),
        "name" : "Sumit",
        "age" : 20,
        "branch" : "CSE",
        "course" : "C++ STL",
        "mode" : "online",
        "paid" : true,
        "amount" : 1499,
        "year" : 2020
}
{
        "_id" : ObjectId("5e54103592e6dfa3fc48ddb1"),
        "name" : "Rohit",
        "age" : 21,
        "branch" : "CSE",
        "course" : "C++ STL",
        "mode" : "online",
        "paid" : true,
        "amount" : 1499
}
> db.student.deleteMany({})
{ "acknowledged" : true, "deletedCount" : 2 }
>
```

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.



Docker Components

These are the Docker Components:

1. 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. We have some commands through which we can communicate the Docker client. Then the docker client passes those commands to the Docker daemon.

docker build ...

docker run ...

docker push ..etc.

2. DockerHost

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

a) 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.

b) 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.
- Run the docker images using the docker run command.
- Push the docker image to the public registry like DockerHub using the **docker push** command

after pushed we can access these images from anywhere using docker pull command.

An image can be used to build a container. Container images can be shared across teams within an enterprise using a private container registry, or shared with the world using a public registry like Docker Hub.

c) Docker Containers:

A container is 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

Through the docker networking, we can communicate one container to other containers.

By default, we get three different networks on the installation of Docker – 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. If we want to persist the container data use the docker storage concept.

We can store data within the writable layer of a container but it requires a storage driver. In terms of persistent storage, Docker offers the following options:

- Data Volumes
- Data-Volume Container
- Bind Mounts

3. 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 following commands

docker push

docker pull

docker run

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

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 — one of these is active and the other environment 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.

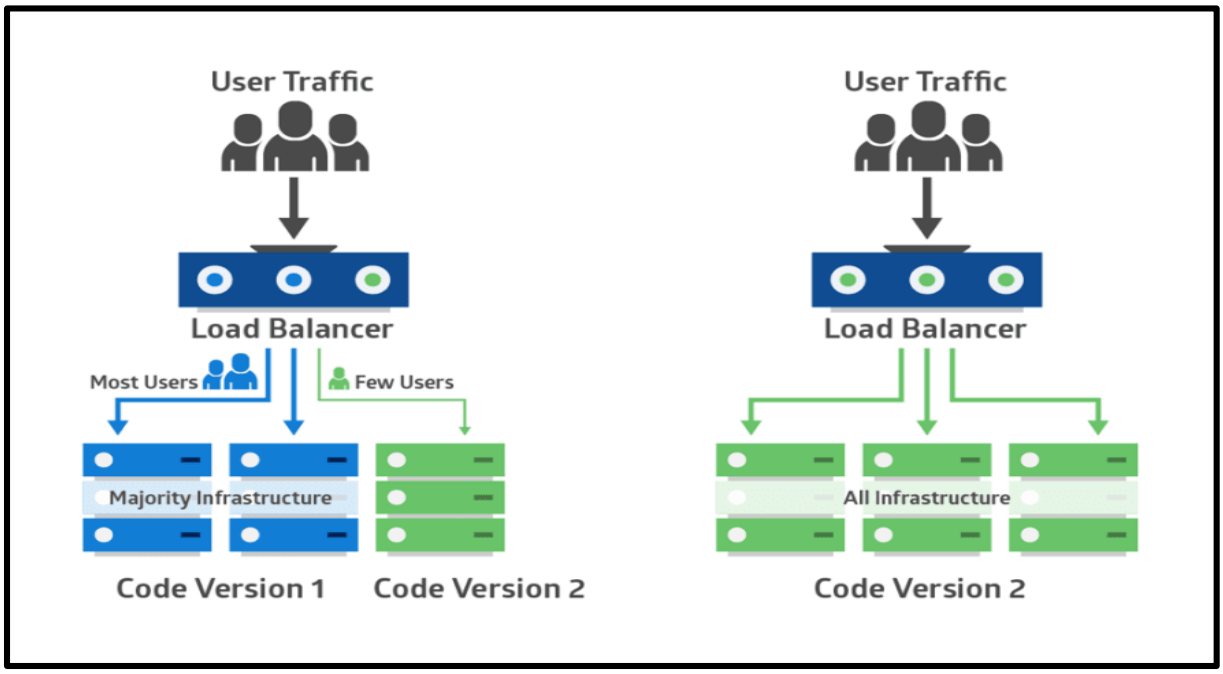
Blue/green deployment provide the following benefits to businesses

- Simple, fast, easy to understand and implement.
- Rollback is straightforward as teams need to simply flip traffic back to the old environment in case any issue arises.
- Blue-green deployments are not as risky and vulnerable to losses as compared to other deployment strategies.

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.

Canary deployments provide the following benefits to businesses.

- Allows enterprises to test in production with real users and use cases.
- Enables comparison of different service versions side by side.
- Cheaper than blue-green deployments because it does not require two production environments.
- DevOps team can rapidly and safely trigger a rollback to a previous version of an application.



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.

VINUTHA G S
Lecturer in CS& E
Government Polytechnic, Ramanagara