**V Semester Diploma Examination MAY-2024**

**Full Stack Development-20CS52I**

**Time:03 Hrs**

**Note: i) Answer one Full question from each section.**

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

**SECTION-I**

1. a) Explain how design thinking has brought revolution in Education System. 10

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

1. a) 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 has brought revolution in digital payments. 10

b) Explain the process of design thinking. Apply the process of design thinking that led to the evolution of smart phones. 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) 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) Compare spring and spring boot. 10

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

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 the rules to follow for an API to the RESTful. 10

**SECTION-IV**

7. a) Compare the databases MySQL and MongoDB 10

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

8. a) How to convert monolithic application to microservice architecture? Explain with an example. 10

b) Implement Inversion of Control (IOC) container in Spring Boot with an example. 10

# SECTION-V

9. a) Create a form to add a new product detail to the product catalogue using React. 10

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

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

b)Draw the CI/CDbuild process flow diagram for an online application and explain each component. 10

**Full Stack Development-20CS52I**

***SCHEME OF EVALUATION***

**Note: i) Answer one Full question from each section.**

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

**SECTION-I**

1. a)Explanation of Design Thinking Related to Education System any 10 points related Education System. **( 1 point=1 Mark)**

b) Explanation**2 Marks+Steps 8 Marks**

2. a) Explanation of Design Thinking.

b) 5 steps of Design Thinking Process- **5 marks**, Smartphone Design Thinking - **5 Marks**

**SECTION-II**

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

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

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

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

**SECTION-III**

1. a) Comparision. **(5 points=10M).**

b)Entity class -**2 Marks**, 4 controller carries 2 marks each -- **4\*2 =8 Marks**

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

b) List rules-**2 Marks** ,Explanation of **4 rules-4\*2=8 Marks**

**SECTION IV**

7. a) Comparision **(5 Points=10 Marks)(1 Point=2 Marks)**

b) Writing program 10Marks,**Entity class 5 Marks** +**Controller Class 5 Marks.**

8. a) coversion steps **2Marks** + steps explanation **8 Marks= 10Marks**

b) Explanation  **2 Marks** +Implementation (Program) **8 Marks=10marks**

**SECTION V**

9. a) Program = **10Marks**

b) Blue Green Deployment Model**-5 Marks** + Canary deployment **5 Marks =10Marks**

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

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

**1. a) Explain how design thinking has brought revolution in Education system 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.

Design thinking has indeed made a significant impact on the education system by revolutionizing the way educators approach teaching and learning. Here are some key ways in which design thinking has brought about positive changes in the education system:

1. \*\*Student-Centered Approach\*\*: Design thinking focuses on understanding the needs and perspectives of the end-users, in this case, the students. By putting students at the center of the education process, educators can design learning experiences that are more engaging, relevant, and effective.

2. \*\*Empathy and Understanding\*\*: Design thinking emphasizes empathy and understanding the needs, motivations, and challenges of students. By empathizing with students, educators can create solutions that truly address their needs and provide a more personalized learning experience.

3. \*\*Iterative and Collaborative Process\*\*: Design thinking encourages an iterative and collaborative approach to problem-solving. Educators can prototype and test new teaching methods, curriculum designs, or learning tools in collaboration with students, colleagues, and other stakeholders, leading to continuous improvement and innovation in education.

4. \*\*Creative Problem-Solving\*\*: Design thinking promotes creative problem-solving by encouraging educators to think outside the box, explore multiple perspectives, and generate innovative solutions to educational challenges. This approach can lead to the development of new teaching methods, interdisciplinary programs, and experiential learning opportunities.

5. \*\*Human-Centered Innovation\*\*: Design thinking helps educators focus on creating human-centered solutions that meet the diverse needs of students. By designing solutions that are user-friendly, engaging, and accessible, educators can create a more inclusive and equitable learning environment.

6. \*\*Real-World Relevance\*\*: Design thinking encourages educators to connect classroom learning to real-world problems and contexts. By engaging students in authentic, project-based learning experiences, educators can help students develop critical thinking, problem-solving, and collaboration skills that are essential for success in the 21st century.

Overall, design thinking has brought a revolution in the education system by promoting a student-centered, empathetic, and innovative approach to teaching and learning. By applying the principles of design thinking, educators can create a more engaging, relevant, and effective learning experience for students, preparing them for success in an increasingly complex and dynamic world.

**1 b) 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.

**2 a) 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.

**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).

**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.

**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.

**2. b) 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.



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.

**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**

## **Sign-up:**

As an unauthorized user,

I want to sign up for the Hotel Booking application through a sign-up form,

So that I can access to book a room.

Acceptance Criteria:

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

3. If I sign up with an incorrect detail which is specified in step1, I will receive an error message for in correct information.

If we are trying to signup with an existing email address, we will receive an error message saying"email exists."

## **2. Login**

As an authorized user, I want to login for Hotel Booking 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 in correct username or password, the normal message will be displayed as "invalid login”.

## **3. Searching a Room**

As an authorized user,I want to search for a room in Hotel Booking 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 anauthorized user,I want to book a room in Hotel Booking application, so that I can reserve the room in a specific location and date.

## AcceptanceCriteria:

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 Hotel Booking application, so that I can prevent unauthorized access of my profile.

**Acceptance Criteria:**When I logout 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 HotelBooking Homepage.

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 thedesign 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 2adults and 2 children and makes a payment(End toEndTest).

7. User make sasuccessful payment for their hotel booking.

8. User makes unsuccessful payment of their hotel booking.

9.Hotel Room Unavailability– User searches for dates that are unavailable and system recommends alternative date and room types.

10.User wants to Amend an existing booking by adding an additional feature (e.g. increase length of stay/addingbreakfast).

11. User cancels their booking and system refunds money–Test Refund Conditions

12.User cancels their booking and system does NOT re fund money–Test Refund Conditions.

13.User wants to make a group booking.

14 .User wants toValidate Booking Page displays correct booking data–Visualcheck

And Confirm Payment Page is displayed when user selects“Make Payment”. End to End Test of Hotel Booking Engine.

**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.**

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

* + **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:**

1. Mobile number linked to bank account should be used for app registration on smart phone.
2. Choose google account associated with user smart phone.
3. Verify the registered mobile number with OTP.
4. 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:**

1. Must add preferred bank name registered with user mobile number.
2. App must find the account number linked to the mobile number for the preferred bank.
3. 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:**

1. User must scan QRcode to transfer money–amount will be automatically displayed or user must enter the amount.
2. User must enter mobile number, amount to pay to his friends/contact.
3. For bank transfer, user must enter account number, IFSCcode, account holder’s name. After successful linking, user can pay by entering desired amount.
4. For bill payment, bill needs to be liked to service providers. Once liked user gets outstanding amount to be paid.
5. 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:**

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

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

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

1. Verify whether user is able to download Gpay app from play store and installation of app on different smart phone like android, ios etcis possible or not.
2. Verify4-digit pin is validated for login authentication.
3. Check whether mobile number is verified with one time password.

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

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

## **Test Cases for the Digital Payment**

1. Ensure camera is opened for scanning when scan QRcode option is selected.
2. Ensure when valid mobile number is entered, authorized gpay recipient name is displayed.
3. Check whether account number and IFSC validation are done for bank transfer.
4. Check whether user is able to access list of bill provider for each category & verify his account linked to the same or not.
5. Ensure user gets acknowledgement for completion of every successful transaction.

## **Test Cases for the Utilities**

1. Check whether user is able to access/apply offer and rewards available on the app.
2. Check whether user is able to retrieve bank balance from bank or not.

**SECTION-III**

**5 a) Compare spring and spring boot.**

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Spring** | **Spring Boot** |
| 1. | Spring is an open-source light weight frame work widely used to develop enterprise applications. | Spring Boot is built on top of the conventional spring framework, widely used to develop RESTAPIs. |
| 2. | The most important feature of the Spring Frame work is dependency injection. | The most important feature of the Spring Boot is Auto configuration. |
| 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 I 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 boiler plate code for smaller tasks. | In Spring Boot, there is reduction in boiler plate code. |
| 9. | Developers have to defined independencies manually in the pom.xml file. | pom.xml file internally handles the required dependencies |

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

## **StudentEntity Class–Student.java**

importjavax.persistence.\*;

@Entity Annotation

publicclass Student {

@Id//Annotation

private intid;privateStringname;

privateStringbranch;

publicStudent() {}//Constructors

public Student(int id, String name, String branch) {this.id =id;

this.name = name;this.branch=branch:}

publicintgetId(){//Getters&Settersmethodsreturnid;}

publicvoidsetId(intid){this.id = id;}

publicStringgetName(){returnname;}

publicvoidsetName(Stringname){this.name=name;}

public String getBranch() {returnfees; }

publicvoidsetBranch(Stringbranch){this.branch=branch; }

}

**StudentRestControllerClass–StudentController.java**

@Rest controller//ControllerAnnotation

Public class Student Controller{

@Autowired//DependencyInjection

Private Student Service student service;//Service Class Object

@GetMapping(“/students”)//DisplayStudent ListpublicList<Student>getAllStudentDetails(){

returnstudentservice.getAllStudentDetails();}

@GetMapping(“/students/{id}”)//FindStudentbyIdpublicList

<Student>getStudentDetails(@PathVariable String id){

returnstudentservice.getStudentDetails(id);}@PostMapping(“/students/”)

//Add Student Detailspublic List<Student> add Student Details(@RequestBodyStudent student){studentservice.addStudentDetails(student);}

@PutMapping(“/students/{id}”)//Update Student Details public List<Student>updateStudentDetails(@RequestBodyStudentstudet,@PathVariableString id){

studentservice.updateStudentDetails(student,id);}

@DeleteMapping(“/students/”)//DeleteStudentDetailspublic List<Student>deleteStudentDetails(@RequestBodyStudent student){studentservice.deleteStudentDetails(student);}

}

**Note**:Code for Student Service class is optional.

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

## **Create Operations**

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

1.db.collection.insertOne()

2.db.collection.insertMany()

## **Insert One()**

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

**Example:** We are considering a collection called Records DB. We can insert a single entry into our collection by calling the insert One() method on Records DB. 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: "8years",species:"Dog",

Address:"Kuvempunagar, Mysore",chipped: true

})

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

db.RecordsDB.insertOne({

...name:"Rakesh",

...age:"10years",

...species:"Cat",

...Address:"Ashok Nagar, Tumkur",

...chipped: true

...})

{

"acknowledged" :true,

"insertedId":ObjectId("5fd989674e6b9ceb8665c31D")

}

## **Insert Many()**

We can insert multiple items at one time by calling the insert Many() method on the desired collection. In this case, we pass multiple items into our chosen collection (Records DB) 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:"Rakesh",age:"10years",species:"Cat",

Address:"Ashok Nagar, Tumkur",

chipped:true},

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

db.RecordsDB.insertMany([{ name: "Rakesh", age: "10 years", species: "Cat",Address: " Ashok nagar,Tumkur ",chipped:true},

{name:"Keerthana", age:"12years",

species:"Cat",Address:"Jayangar, Bangalore",chipped:true}])

{

"acknowledged":true,"insertedIds": [ObjectId("5fd98ea9ce6e8850d88270b4"),ObjectId("5fd98ea9ce6e8850d88270b5")

]

}

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

For an API to be REST ful 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

## **Uniform Interface**

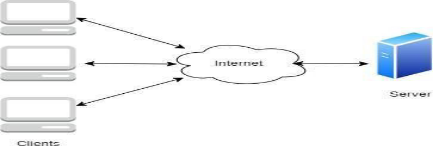
The API should facilitate communication between the client and server as they exchange data. To efficiently exchange data, we need a uniform interface.

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

## **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 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 one ascend change.



## 

## **Stateless**

It’s important that each end point 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 every thing 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.

## **Cacheable**

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

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

## **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 Gate way.

## **Code on Demand**

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

**SECTION-IV**

**7.a) Compare the databases MySQL and MongoDB 10M**

|  |  |  |
| --- | --- | --- |
| **Comparison Basis** | **MySQL** | **MongoDB** |
| **Definition** | It is an open-source, cross-platform relational database management system built by Swedish Company MYSQL AB and currently supported by the Oracle. | It is a popular open-source NoSQL database management system developed and owned by MongoDB Inc. that stores data in JSON-like format. |
| **Release** | It was released on 23 May 1995. | It was released on 11 February 2009. |
| **Written in** | It is written in C and C++. | It is written in C, C++, and Java. |
| **Database Structure** | MySQL stores each individual records in tables and can access it by using the SQL queries. | MongoDB stores each individual record in JSON-like documents that may vary in structures. |
| **SQL or NoSQL** | MySQL uses Structured Query Language to process and access the database. We cannot change its schema. The inputs can only enter with a defined schema. SQL does not allow to work with unstructured and semi-structured data. | MongoDB is a NoSQL database system. It means we can define and adhere to the predefined structure of the incoming data. NoSQL allows working with unstructured and semi-structured data, which is not possible in RDBMS. Its schema can be changed. |
| **Queries Differences** | To select all records, it uses:   * Select \* from table\_name;   To insert records:   * INSERT INTO table\_name(stud\_id, branch, state) VALUES ('Joel01', 'MTech', 'Capetown') | To select all records, it uses:   * db.customer.find();   To insert records:   * db.table\_name.insert({ stud\_id: 'Joel01', branch: 'MTech', state: 'Capetown' }) |

**7 b) RESTful API supports CRUD operations. Design Entity class and Controller class.**

Create an Entity class for the Employee, a Controller class to handle HTTP requests, and a Repository to interact with the database.

**Entity Class (Employee.java):**

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

**@Entity**

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String employeeName;

private String department;

// Constructors, getters, and setters

// Constructor without id for creating new employees

public Employee(String employeeName, String department) {

this.employeeName = employeeName;

this.department = department;

}

// Default constructor for JPA

public Employee() {

}

// Getters and setters

}

**Repository Interface (EmployeeRepository.java):**

import org.spring framework.data.jpa.repository.JpaRepository;

import org.spring framework.stereotype.Repository;

@Repository

public interface EmployeeRepository extends JpaRepository<Employee, Long> {

}

**Controller Class (EmployeeController.java):**

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.\*;

import java.util.List;

@RestController

@RequestMapping("/employees")

public class EmployeeController {

private final EmployeeRepository employeeRepository;

@Autowired

public EmployeeController(EmployeeRepository employeeRepository) {

this.employeeRepository = employeeRepository;

}

// Endpoint to get all employees

@GetMapping

public ResponseEntity<List<Employee>> getAllEmployees() {

List<Employee> employees = employeeRepository.findAll();

return new ResponseEntity<>(employees, HttpStatus.OK);

}

// Endpoint to get an employee by id

@GetMapping("/{id}")

public ResponseEntity<Employee> getEmployeeById(@PathVariable Long id) {

Employee employee = employeeRepository.findById(id).orElse(null);

if (employee == null) {

return new ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

return new ResponseEntity<>(employee, HttpStatus.OK);

}

// Endpoint to create a new employee

@PostMapping

public ResponseEntity<Employee> createEmployee(@RequestBody Employee employee) {

Employee createdEmployee = employeeRepository.save(employee);

return new ResponseEntity<>(createdEmployee, HttpStatus.CREATED);

}

// Endpoint to update an existing employee

@PutMapping("/{id}")

public ResponseEntity<Employee> updateEmployee(@PathVariable Long id, @RequestBody Employee employee) {

Employee existingEmployee = employeeRepository.findById(id).orElse(null);

if (existingEmployee == null) {

return new ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

existingEmployee.setEmployeeName(employee.getEmployeeName());

existingEmployee.setDepartment(employee.getDepartment());

employeeRepository.save(existingEmployee);

return new ResponseEntity<>(existingEmployee, HttpStatus.OK);

}

// Endpoint to delete an employee by id

@DeleteMapping("/{id}")

public ResponseEntity<Void> deleteEmployee(@PathVariable Long id) {

Employee employee = employeeRepository.findById(id).orElse(null);

if (employee == null) {

return new ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

employeeRepository.delete(employee);

return new ResponseEntity<>(HttpStatus.NO\_CONTENT);

}

}

**Spring Boot Application (EmployeeManagementApplication.java):**

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class EmployeeManagementApplication {

public static void main(String[] args) {

SpringApplication.run(EmployeeManagementApplication.class, args);

}

}

**8.a) How to convert monolithic application to microservice architecture? Explain with an example. 10M**

A typical process to migrate from a monolithic system to a microservices-based system involves the following steps:

1. Identify logical components.

2. Flatten and refactor components.

3. Identify component dependencies.

4. Identify component groups.

5. Create an API for remote user interface.

6. Migrate component groups to macro services (move component groups to separate projects and make separate deployments).

7. Migrate macro services to microservices.

8. Repeat steps 6-7 until complete.

**1.Identify Logical Components**

There are three main information components with the data used in the system:

\*Data objects

\*Data actions

\*Job to perform and use cases

The data objects are the logical constructs representing the data being used. The data actions are the commands that are used on one or more data objects, possibly on different types of data, to perform a task. The job to perform represents the function the users are calling to fulfill their organizational roles. The jobs to perform may be captured as use cases, user stories, or other documentation involving user input.

**Example: Movie application**

Monolith Architecture

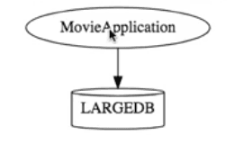
The salient features of monolith applications are:

* Released, or taken to production, once every few week or months or years
* Generally have a wide range of features and functionality
* Have a development team of over 50 people working on them
* Debugging problems that arise in them, is a huge challenge

It is almost impossible to bring in new technologies and technical processes, midway through the lifetime of such an application.

Monolith applications are typically huge, with them having a million lines of code on average.

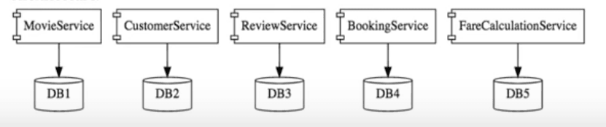
A monolithic application looks as follows:



We have a large application talking to a large database. Micro services Architecture

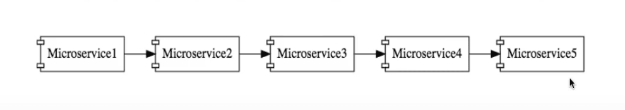
In micro services architectures, instead of building a large application, we build a number of smaller micro services.

This is how we would split up the monolith Movie Application into micro service application



As we can see the databases are also separated out.

Micro services architecture involves a number of small, well-designed micro services, that exchange messages among themselves.



8 b) Implement Inversion of Control (loC) containers in Spring Boot with an example.10M

# Spring IoC (Inversion of Control) Container is the core of [Spring Framework](https://www.geeksforgeeks.org/introduction-to-spring-framework/). It creates the objects, configures and assembles their dependencies, manages their entire life cycle. The Container uses Dependency Injection(DI) to manage the components that make up the application. It gets the information about the objects from a configuration file(XML) or Java Code or Java Annotations and Java POJO class. These objects are called Beans. Since the Controlling of Java objects and their lifecycle is not done by the developers, hence the name Inversion Of Control.

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

**import**org.springframework.context.ApplicationContext;

**import**org.springframework.context.support.ClassPathXmlApplicationContext;

**publicclass**Mobile {

**publicstaticvoid**main(String[] args) {

        // Using ApplicationContext tom implement Spring IoC

        ApplicationContext applicationContext = **new**ClassPathXmlApplicationContext("beans.xml");

        // Get the bean

        Sim sim = applicationContext.getBean("sim", Sim.**class**);

        // Calling the methods

        sim.calling();

        sim.data();

    }

}

# Out put :

Airtel Calling

Airtel Data

# SECTION-V

9.a) Create a form to add a new product detail to the product catalogue using React.

**React component code for Productform.js**

import React, { useState } from 'react';

const ProductForm = () => {

const [productName, setProductName] = useState('');

const [description, setDescription] = useState('');

const [price, setPrice] = useState('');

const handleSubmit = (event) => {

event.preventDefault();

// Assuming you'll perform validation here before proceeding

// For simplicity, we'll just log the product details to the console.

console.log('Product Name:', productName);

console.log('Description:', description);

console.log('Price:', price);

// Add code here to submit the data to the server or state management system.

// You can also reset the form fields after submission if needed.

};

return (

<form onSubmit={handleSubmit}>

<div>

<label htmlFor="productName">Product Name:</label>

<input

type="text"

id="productName"

value={productName}

onChange={(e) => setProductName(e.target.value)}

required

/>

</div>

<div>

<label htmlFor="description">Description:</label>

<textarea

id="description"

value={description}

onChange={(e) => setDescription(e.target.value)}

required

/>

</div>

<div>

<label htmlFor="price">Price:</label>

<input

type="number"

id="price"

value={price}

onChange={(e) => setPrice(e.target.value)}

required

/>

</div>

<button type="submit">Add Product</button>

</form>

);

};

export default Product Form;

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

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.

Roll back is straight forward 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 the 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 sub set of users before making the software functional for everyone.

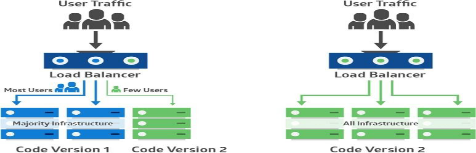
Canary deployments provide the following benefits to businesses.

\*Allows enterprises to test in production with real users and use cases.

\*Enables comparison of different service versions side by side.

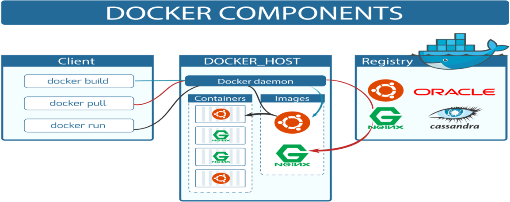
\*Cheaper than blue-green deployments because it does not require two production environments.

\*DevOps team can rapidly and safely trigger a rollback to a previous version of an application.



**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. DOCKERCLIENT

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 **dockerdaemon**,which carries the mout. 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 theDocker daemon.

Docker build ...docke run...docker push..etc.

## 2. 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

Docker Daemon is a persistent background process that manages Docker images,containers,networks,and storage volumes.TheDocker 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 meta data 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 Docker Hubusing the **docker push** command after pushed we can access these images from any where using docker pull command.

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

## c) Docker Containers:

A containeris 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 an 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. Then one 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 when ever we remove or kill the container the nall 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 with in the writable layer of a container butit requires a storage driver. Interms of persistent storage,Docker offers the following options:

\*DataVolumes

\*Data-VolumeContainer

\*BindMounts

## 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

**10 b) Draw the CI/CD build process flow diagram for an online application and explain each component.**

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.

Automated Build

(Build Server)

(Artifact)

Continuous Integration

(CI server triggered)

Source Code

(Version Control)

Automated Unit Tests

and Code Analysis

(Test Results )

Automated Testing

Environment

(Test Results )

Deployment to

Staging Environment

(Deployment)

Deployment to

Production

User Acceptance

Testing (UAT)

(Approval )

**1. 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.

**2. 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.

**3. 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.

**4. 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.

**5. 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.

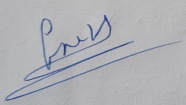
**6. Deployment to Staging Environment:** If all the previous stages (build and automated tests) are successful, the application is deployed to a staging environment. The staging environment is a near-production replica where final testing is conducted before going live.

**7. 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.

***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.



**Harish G. N**

**Lecturer in CS& E**

**Government Polytechnic,Chitradurga**