Sainsbury's Technical Assessment Prep:

**1.) React Project: Tic-Tac-Toe Game:**

Code (in App.js) - The details should be in the question:

import React, { useState } from 'react';

import './App.css'; // Ensure you have some basic styling for the classes mentioned

// Square Component

function Square({ value, onClick }) {

return (

<button className="square" onClick={onClick}>

{value}

</button>

);

}

// Board Component

function Board() {

const [squares, setSquares] = useState(Array(9).fill(null));

const [xIsNext, setXIsNext] = useState(true);

const [status, setStatus] = useState('Next player: X');

const handleClick = (i) => {

const newSquares = squares.slice();

if (calculateWinner(squares) || newSquares[i]) {

return;

}

newSquares[i] = xIsNext ? 'X' : 'O';

setSquares(newSquares);

const winner = calculateWinner(newSquares);

if (winner) {

setStatus(`Winner: ${winner}`);

} else if (newSquares.every(Boolean)) {

setStatus('Tie');

} else {

setStatus(`Next player: ${xIsNext ? 'O' : 'X'}`);

}

setXIsNext(!xIsNext);

};

const resetGame = () => {

setSquares(Array(9).fill(null));

setXIsNext(true);

setStatus('Next player: X');

};

return (

<div>

<div className="status">{status}</div>

<div className="board-row">

<Square value={squares[0]} onClick={() => handleClick(0)} />

<Square value={squares[1]} onClick={() => handleClick(1)} />

<Square value={squares[2]} onClick={() => handleClick(2)} />

</div>

<div className="board-row">

<Square value={squares[3]} onClick={() => handleClick(3)} />

<Square value={squares[4]} onClick={() => handleClick(4)} />

<Square value={squares[5]} onClick={() => handleClick(5)} />

</div>

<div className="board-row">

<Square value={squares[6]} onClick={() => handleClick(6)} />

<Square value={squares[7]} onClick={() => handleClick(7)} />

<Square value={squares[8]} onClick={() => handleClick(8)} />

</div>

<button className="reset" onClick={resetGame}>Reset</button>

</div>

);

}

// Helper Function to Calculate the Winner

function calculateWinner(squares) {

const lines = [

[0, 1, 2],

[3, 4, 5],

[6, 7, 8],

[0, 3, 6],

[1, 4, 7],

[2, 5, 8],

[0, 4, 8],

[2, 4, 6],

];

for (let i = 0; i < lines.length; i++) {

const [a, b, c] = lines[i];

if (squares[a] && squares[a] === squares[b] && squares[a] === squares[c]) {

return squares[a];

}

}

return null;

}

// App Component

export default function App() {

return (

<div className="game">

<div className="game-board">

<Board />

</div>

</div>

);

}

App.css:

.status {

margin-bottom: 10px;

}

.square {

width: 100px;

height: 100px;

background: #fff;

border: 1px solid #999;

float: left;

font-size: 24px;

font-weight: bold;

}

.board-row:after {

clear: both;

content: "";

display: table;

}

.reset {

margin-top: 20px;

padding: 10px;

background-color: #f00;

color: #fff;

border: none;

cursor: pointer;

}

**2.) Java (presumably) with SpringBoot. Create a REST API:**

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

1. Next step is the Main Application Class:

DemoApplication.java:

package com.example.demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class DemoApplication {

public static void main(String[] args) {

SpringApplication.run(DemoApplication.class, args);

}

}

1. Create User.java:

package com.example.demo.model;

public class User {

private Long id;

private String name;

private String email;

// Constructors, getters, and setters

public User() {}

public User(Long id, String name, String email) {

this.id = id;

this.name = name;

this.email = email;

}

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

}

1. Create User Controller (UserController.java):

package com.example.demo.controller;

import com.example.demo.model.User;

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

import java.util.ArrayList;

import java.util.List;

@RestController

@RequestMapping("/api/users")

public class UserController {

private List<User> users = new ArrayList<>();

@GetMapping

public List<User> getAllUsers() {

return users;

}

@GetMapping("/{id}")

public User getUserById(@PathVariable Long id) {

return users.stream().filter(user -> user.getId().equals(id)).findFirst().orElse(null);

}

@PostMapping

public User createUser(@RequestBody User user) {

users.add(user);

return user;

}

@PutMapping("/{id}")

public User updateUser(@PathVariable Long id, @RequestBody User userDetails) {

User user = users.stream().filter(u -> u.getId().equals(id)).findFirst().orElse(null);

if (user != null) {

user.setName(userDetails.getName());

user.setEmail(userDetails.getEmail());

}

return user;

}

@DeleteMapping("/{id}")

public void deleteUser(@PathVariable Long id) {

users.removeIf(user -> user.getId().equals(id));

}

}

1. Run the Spring Boot application using:

‘mvn spring-boot:run’

(test it via Postman)

A screenshot of a computer

Description automatically generated

**IF THERE IS MENTION OF THE CREDIT CARD FEATURE, USE THE ONE BELOW:**

**A screenshot of a computer program

Description automatically generated**

ii) Implement the main application class:

**package com.example.creditcard;**

**import org.springframework.boot.SpringApplication;**

**import org.springframework.boot.autoconfigure.SpringBootApplication;**

**@SpringBootApplication**

**public class CreditCardApplication {**

**public static void main(String[] args) {**

**SpringApplication.run(CreditCardApplication.class, args);**

**}**

**}**

iii) Implement the Model:

**package com.example.creditcard.model;**

**public class CreditCardRequest {**

**private String cardNumber;**

**// Getters and setters**

**public String getCardNumber() {**

**return cardNumber;**

**}**

**public void setCardNumber(String cardNumber) {**

**this.cardNumber = cardNumber;**

**}**

**}**

**iv)** Implement the Service Layer:

**package com.example.creditcard.service;**

**import org.springframework.stereotype.Service;**

**@Service**

**public class CreditCardService {**

**public boolean isValidCreditCard(String cardNumber) {**

**int nDigits = cardNumber.length();**

**int nSum = 0;**

**boolean isSecond = false;**

**for (int i = nDigits - 1; i >= 0; i--) {**

**int d = cardNumber.charAt(i) - '0';**

**if (isSecond) {**

**d = d \* 2;**

**}**

**nSum += d / 10;**

**nSum += d % 10;**

**isSecond = !isSecond;**

**}**

**return (nSum % 10 == 0);**

**}**

**}**

**vi)** Implement the controller layer:

**package com.example.creditcard.controller;**

**import com.example.creditcard.model.CreditCardRequest;**

**import com.example.creditcard.service.CreditCardService;**

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

**import org.springframework.http.ResponseEntity;**

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

**@RestController**

**@RequestMapping("/api/creditcard")**

**public class CreditCardController {**

**@Autowired**

**private CreditCardService creditCardService;**

**@PostMapping("/validate")**

**public ResponseEntity<String> validateCreditCard(@RequestBody CreditCardRequest request) {**

**boolean isValid = creditCardService.isValidCreditCard(request.getCardNumber());**

**if (isValid) {**

**return ResponseEntity.ok("Valid credit card number");**

**} else {**

**return ResponseEntity.badRequest().body("Invalid credit card number");**

**}**

**}**

**}**

vii) Implement Unit Testing:

**package com.example.creditcard;**

**import com.example.creditcard.service.CreditCardService;**

**import org.junit.jupiter.api.Test;**

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

**import org.springframework.boot.test.context.SpringBootTest;**

**import static org.junit.jupiter.api.Assertions.assertFalse;**

**import static org.junit.jupiter.api.Assertions.assertTrue;**

**@SpringBootTest**

**class CreditCardApplicationTests {**

**@Autowired**

**private CreditCardService creditCardService;**

**@Test**

**void testValidCreditCard() {**

**String validCardNumber = "4539578763621486"; // Example of a valid card number**

**assertTrue(creditCardService.isValidCreditCard(validCardNumber));**

**}**

**@Test**

**void testInvalidCreditCard() {**

**String invalidCardNumber = "4539578763621487"; // Example of an invalid card number**

**assertFalse(creditCardService.isValidCreditCard(invalidCardNumber));**

**}**

**}**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

1. **To calculate chess pawn movement across a looping chessboard using Java:**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Code in Java:

public class LoopingChessboard {

private static final int BOARD\_SIZE = 8;

// Method to calculate new position for white pawn

public static int[] moveWhitePawn(int x, int y) {

int newX = (x - 1 + BOARD\_SIZE) % BOARD\_SIZE;

return new int[]{newX, y};

}

// Method to calculate new position for black pawn

public static int[] moveBlackPawn(int x, int y) {

int newX = (x + 1) % BOARD\_SIZE;

return new int[]{newX, y};

}

public static void main(String[] args) {

// Initial positions

int[] whitePawnPosition = {6, 4}; // e2 in traditional notation

int[] blackPawnPosition = {1, 4}; // e7 in traditional notation

// Move the pawns

System.out.println("Initial White Pawn Position: (" + whitePawnPosition[0] + ", " + whitePawnPosition[1] + ")");

whitePawnPosition = moveWhitePawn(whitePawnPosition[0], whitePawnPosition[1]);

System.out.println("New White Pawn Position: (" + whitePawnPosition[0] + ", " + whitePawnPosition[1] + ")");

System.out.println("Initial Black Pawn Position: (" + blackPawnPosition[0] + ", " + blackPawnPosition[1] + ")");

blackPawnPosition = moveBlackPawn(blackPawnPosition[0], blackPawnPosition[1]);

System.out.println("New Black Pawn Position: (" + blackPawnPosition[0] + ", " + blackPawnPosition[1] + ")");

}

}

A white text on a white background

Description automatically generated

1. **React Test: Sainsbury Front-End Test:**

“In this exercise you will develop a product listing page and shopping basket.”

**Version 1: Basic Styling:**

1. App.js:

import React, { useState, useEffect } from 'react';

import './App.css';

import ProductList from './components/ProductList';

import Basket from './components/Basket';

function App() {

const [products, setProducts] = useState([]);

const [basket, setBasket] = useState([]);

useEffect(() => {

fetch('https://jsainsburyplc.github.io/front-end-test/products.json')

.then(response => response.json())

.then(data => setProducts(data))

.catch(error => console.error('Error fetching products:', error));

}, []);

const addToBasket = (product) => {

setBasket([...basket, product]);

};

return (

<div className="App">

<h1>Product Listing</h1>

<Basket basket={basket} />

<ProductList products={products} addToBasket={addToBasket} />

</div>

);

}

export default App;

1. ProductList.js:

import React from 'react';

import Product from './Product';

function ProductList({ products, addToBasket }) {

return (

<div className="product-list">

{products.map(product => (

<Product key={product.sku} product={product} addToBasket={addToBasket} />

))}

</div>

);

}

export default ProductList;

1. Product.js:

import React from 'react';

function Product({ product, addToBasket }) {

return (

<div className="product">

<img src={product.image} alt={product.title} />

<h2>{product.title}</h2>

<p>{product.sku}</p>

<p>Price: {product.price}</p>

<button onClick={() => addToBasket(product)}>Add to Basket</button>

</div>

);

}

export default Product;

1. Basket.js:

import React from 'react';

function Basket({ basket }) {

return (

<div className="basket">

<h2>Your Basket ({basket.length} items)</h2>

{basket.map((product, index) => (

<div key={index} className="basket-item">

<p>{product.title}</p>

<p>{product.price}</p>

</div>

))}

</div>

);

}

export default Basket;

1. App.css:

.App {

font-family: Arial, sans-serif;

margin: 20px;

}

.product-list {

display: flex;

flex-wrap: wrap;

gap: 20px;

}

.product {

border: 1px solid #ddd;

padding: 20px;

width: 200px;

text-align: center;

}

.product img {

width: 100%;

height: auto;

}

.basket {

margin-bottom: 20px;

}

.basket-item {

border: 1px solid #ddd;

padding: 10px;

margin-bottom: 10px;

}

**Version 2: More Styling:**

1. App.js:

import React, { useState, useEffect } from 'react';

import './App.css';

import ProductList from './components/ProductList';

import Basket from './components/Basket';

import { FaShoppingBasket } from 'react-icons/fa';

function App() {

// State to hold the list of products

const [products, setProducts] = useState([]);

// State to hold the products added to the basket

const [basket, setBasket] = useState([]);

// State to toggle the basket view

const [showBasket, setShowBasket] = useState(false);

// Fetch products from the API when the component mounts

useEffect(() => {

fetch('https://jsainsburyplc.github.io/front-end-test/products.json')

.then(response => response.json())

.then(data => setProducts(data))

.catch(error => console.error('Error fetching products:', error));

}, []);

// Function to add a product to the basket

const addToBasket = (product) => {

setBasket([...basket, product]);

};

// Function to toggle the basket view

const toggleBasket = () => {

setShowBasket(!showBasket);

};

return (

<div className="App">

<header className="App-header">

<h1>Product Listing</h1>

<button className="basket-icon" onClick={toggleBasket}>

<FaShoppingBasket /> <span>{basket.length}</span>

</button>

</header>

{showBasket ? (

<Basket basket={basket} toggleBasket={toggleBasket} />

) : (

<ProductList products={products} addToBasket={addToBasket} />

)}

</div>

);

}

export default App;

1. ProductList.js:

import React from 'react';

import Product from './Product';

// Component to render a list of products

function ProductList({ products, addToBasket }) {

return (

<div className="product-list">

{products.map(product => (

<Product key={product.sku} product={product} addToBasket={addToBasket} />

))}

</div>

);

}

export default ProductList;

1. Product.js:

import React from 'react';

// Component to render a single product

function Product({ product, addToBasket }) {

return (

<div className="product">

<img src={product.image} alt={product.title} />

<h2>{product.title}</h2>

<p>{product.sku}</p>

<p>Price: {product.price}</p>

<button onClick={() => addToBasket(product)}>Add to Basket</button>

</div>

);

}

export default Product;

1. Basket.js:

import React from 'react';

// Component to render the basket

function Basket({ basket, toggleBasket }) {

return (

<div className="basket">

<h2>Your Basket ({basket.length} items)</h2>

<button onClick={toggleBasket}>Close</button>

{basket.map((product, index) => (

<div key={index} className="basket-item">

<img src={product.image} alt={product.title} className="basket-item-image" />

<div className="basket-item-details">

<p>{product.title}</p>

<p>{product.price}</p>

</div>

</div>

))}

</div>

);

}

export default Basket;

1. App.css:

.App {

font-family: Arial, sans-serif;

margin: 20px;

}

.App-header {

display: flex;

justify-content: space-between;

align-items: center;

}

.product-list {

display: flex;

flex-wrap: wrap;

gap: 20px;

}

.product {

border: 1px solid #ddd;

padding: 20px;

width: 200px;

text-align: center;

}

.product img {

width: 100%;

height: auto;

}

.basket {

margin-bottom: 20px;

}

.basket-item {

display: flex;

align-items: center;

border: 1px solid #ddd;

padding: 10px;

margin-bottom: 10px;

}

.basket-item-image {

width: 50px;

height: auto;

margin-right: 10px;

}

.basket-item-details {

display: flex;

flex-direction: column;

}

.basket-icon {

background: none;

border: none;

cursor: pointer;

font-size: 24px;

position: relative;

}

.basket-icon span {

background-color: red;

border-radius: 50%;

color: white;

padding: 2px 6px;

position: absolute;

top: -10px;

right: -10px;

}

1. **Chess-Piece Movement Project/Challenge in React:**
2. App.js:

import React, { useState } from 'react';

import './App.css';

import Chessboard from './components/Chessboard';

function App() {

return (

<div className="App">

<h1>Chess Pawn Movement</h1>

<Chessboard />

</div>

);

}

export default App;

1. Chessboard.js (in components):

import React, { useState } from 'react';

import Square from './Square';

const BOARD\_SIZE = 8;

function Chessboard() {

// State to hold the pawn's position

const [pawnPosition, setPawnPosition] = useState({ row: 6, col: 0 });

// Function to move the pawn up

const movePawnUp = () => {

setPawnPosition(prevPosition => {

const newRow = (prevPosition.row - 1 + BOARD\_SIZE) % BOARD\_SIZE;

return { row: newRow, col: prevPosition.col };

});

};

// Function to move the pawn left

const movePawnLeft = () => {

setPawnPosition(prevPosition => {

const newCol = (prevPosition.col - 1 + BOARD\_SIZE) % BOARD\_SIZE;

return { row: prevPosition.row, col: newCol };

});

};

// Function to move the pawn right

const movePawnRight = () => {

setPawnPosition(prevPosition => {

const newCol = (prevPosition.col + 1) % BOARD\_SIZE;

return { row: prevPosition.row, col: newCol };

});

};

// Generate the board with the pawn's current position

const board = Array.from({ length: BOARD\_SIZE }, (\_, rowIndex) =>

Array.from({ length: BOARD\_SIZE }, (\_, colIndex) => (

<Square

key={`${rowIndex}-${colIndex}`}

isPawn={rowIndex === pawnPosition.row && colIndex === pawnPosition.col}

/>

))

);

return (

<div>

<div className="info">

<p>Pawn Position: Row {pawnPosition.row + 1}, Column {pawnPosition.col + 1}</p>

<button onClick={movePawnUp}>Move Pawn Up</button>

<button onClick={movePawnLeft}>Move Pawn Left</button>

<button onClick={movePawnRight}>Move Pawn Right</button>

</div>

<div className="chessboard">

{board}

</div>

</div>

);

}

export default Chessboard;

1. Square.js:

import React from 'react';

function Square({ value }) {

return (

<div className={`square ${value ? 'filled' : ''}`}>

{value}

</div>

);

}

export default Square;

1. App.css:

.App {

font-family: Arial, sans-serif;

margin: 20px;

text-align: center;

}

.chessboard {

display: grid;

grid-template-columns: repeat(8, 50px);

grid-template-rows: repeat(8, 50px);

margin: 0 auto;

gap: 2px;

}

.square {

width: 50px;

height: 50px;

border: 1px solid #333;

display: flex;

align-items: center;

justify-content: center;

}

.square.filled {

background-color: #333;

color: white;

}