|  |
| --- |
| **CS-4032: Web Programming(BSCS-A,B)** |
| Date: 4th Jun, 2024 |
| **Course Instructor(s)** |
| Mr. Aqib Rehman |

|  |
| --- |
| **Final Exam** |
| **Total Time: 3 Hours** |
| **Total Marks: 155** |
|  |

|  |
| --- |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Student Name Roll No. Course Section Student Signature |

**Do not write anything on the question paper except the information required above.**

**Instructions:**

1. Read the question carefully, understand the question, and then attempt your answers in the provided answer booklet **in the given questions order.**
2. Verify that you have **Ten (10)** printed page of the question paper including this page. There are **Three (3)** questions.
3. This exam is **closed book**. Mobiles, Internet and note-sharing is not allowed. Please see that the area in your threshold is free of any material classified as useful in the paper, i.e. mobile/internet or else there may be a charge of cheating.
4. Read the questions carefully for clarity of context and understanding of meaning and make assumptions wherever required, for neither the invigilator will address your queries, nor the teacher/examiner will come to the examination hall for any assistance.
5. Use only your own stationary.
6. Use only permanent ink-pens. Only the questions attempted with permanent ink-pens will be considered. Any part done using soft pencil cannot be claimed for checking/rechecking.

**Q1: Objectives [60 Marks]**

1. **Give the answers in the table at the end of mcqs on question paper.**
2. **Which HTML attribute is used to define inline styles?**
   * a) style
   * b) class
   * c) styles
   * d) font
3. **What is the correct HTML element for the largest heading?**
   * a) <h1>
   * b) <h6>
   * c) <head>
   * d) <header>
4. **Which HTML attribute is used to provide a unique identifier for an element?**
   * a) id
   * b) class
   * c) name
   * d) tag
5. **What does the <ol> tag represent in HTML?**
   * a) Ordered List
   * b) Object List
   * c) Original List
   * d) Outer List
6. **Which property is used to change the background color in CSS?**
   * a) background-color
   * b) bgcolor
   * c) color
   * d) background
7. **How do you select an element with id "main" in CSS?**
   * a) #main
   * b) .main
   * c) main
   * d) \*main
8. **Which CSS property controls the text size?**
   * a) font-size
   * b) text-size
   * c) font-style
   * d) text-style
9. **Which property is used to change the font of an element?**
   * a) font-family
   * b) font-weight
   * c) font-style
   * d) font-type
10. **Which Bootstrap class provides a responsive fixed-width container?**
    * a) .container
    * b) .container-fluid
    * c) .fixed-container
    * d) .row
11. **How do you create a responsive grid system in Bootstrap?**
    * a) Using rows and columns
    * b) Using flexbox
    * c) Using tables
    * d) Using divs
12. **What is the correct syntax to create a function in JavaScript?**
    * a) function myFunction() {}
    * b) def myFunction() {}
    * c) create myFunction() {}
    * d) method myFunction() {}
13. **How do you write "Hello World" in an alert box?**
    * a) alert("Hello World");
    * b) msg("Hello World");
    * c) msgBox("Hello World");
    * d) alertBox("Hello World");
14. **What will the following code output? console.log(typeof []);**
    * a) object
    * b) array
    * c) list
    * d) undefined
15. **What is the correct syntax to create a React component?**
    * a) class MyComponent extends React.Component {}
    * b) component MyComponent extends React.Component {}
    * c) function MyComponent extends React.Component {}
    * d) class MyComponent extends React.Element {}
16. **How do you pass data from parent to child component in React?**
    * a) Via props
    * b) Via state
    * c) Via context
    * d) Via hooks
17. **Which method in React is used to update the state of a component?**
    * a) this.setState
    * b) this.updateState
    * c) this.changeState
    * d) this.modifyState
18. **What is the purpose of keys in React?**
    * a) To help identify which items have changed, are added, or removed
    * b) To uniquely identify elements for accessibility purposes
    * c) To pass down values to components
    * d) To store component state
19. **Which command is used to install a package globally in Node.js?**
    * a) npm install -g package-name
    * b) npm install package-name
    * c) npm install global package-name
    * d) npm install -global package-name
20. **What is the default port for a Node.js application?**
    * a) 3000
    * b) 8080
    * c) 80
    * d) 5000
21. **How do you create both client and server in Node.js?**
    * a) Using the http module
    * b) Using the net module
    * c) Using the express module
    * d) Using the url module
    * e) Using the dns module
22. **Which method is used to retrieve all documents from a MongoDB collection?**
    * a) find()
    * b) getAll()
    * c) retrieve()
    * d) fetch()
23. **What does CRUD stand for in the context of MongoDB?**
    * a) Create, Read, Update, Delete
    * b) Connect, Read, Update, Delete
    * c) Create, Retrieve, Update, Delete
    * d) Connect, Retrieve, Update, Delete
24. **How do you create a new Express application?**
    * a) var app = express()
    * b) var app = new express()
    * c) var app = Express()
    * d) var app = new Express()
25. **Which method is used to define a GET route in Express?**
    * a) app.get()
    * b) app.route()
    * c) app.fetch()
    * d) app.retrieve()
26. **How do you start a server in Express?**
    * a) app.listen()
    * b) app.start()
    * c) app.run()
    * d) app.launch()
27. **Which middleware function is used to parse incoming request bodies in Express?**
    * a) body-parser
    * b) json-parser
    * c) urlencoded-parser
    * d) data-parser
28. **How do you connect a MongoDB database to a Node.js application using Mongoose?**
    * a) mongoose.connect()
    * b) mongoose.bind()
    * c) mongoose.link()
    * d) mongoose.attach()
29. **Which of the following is true about React components?**
    * a) They can be stateful or stateless.
    * b) They must always be stateful.
    * c) They cannot hold any state.
    * d) They must always be class-based.
30. **Which CSS framework is designed to be used with React?**
    * a) Bootstrap
    * b) Tailwind CSS
    * c) Material-UI
    * d) Foundation
31. **What command is used to create a new React application using Create React App?**
    * a) npx create-react-app my-app
    * b) npm create-react-app my-app
    * c) npx react-create-app my-app
    * d) npm create-app my-app
32. **Which HTTP method is typically used to update data on a server?**
    * a) PUT
    * b) GET
    * c) POST
    * d) DELETE
33. **How do you make a JSON response in an Express route?**
    * a) res.json()
    * b) res.send()
    * c) res.return()
    * d) res.end()
34. **In MongoDB, which of the following is not a valid collection operation?**
    * a) db.collection.drop()
    * b) db.collection.rename()
    * c) db.collection.update()
    * d) db.collection.copy()
35. **Which command is used to initialize a new Node.js project?**
    * a) npm init
    * b) npm start
    * c) npm new
    * d) npm create
36. **Which of the following is not a valid JavaScript data type?**
    * a) Undefined
    * b) Null
    * c) Number
    * d) Character
37. **Which method is used to add one or more elements to the start of an array and returns the new length of the array?**

* a) push()
* b) pop()
* c) concat()
* d) unshift()

1. **How can you create an array in JavaScript?**

* a) var arr = [];
* b) var arr = Array.create();
* c) var arr = new Array();
* d) Both a and c
* e) All

1. **Which statement is used to handle runtime errors in JavaScript?**

* a) try...catch
* b) exception...handle
* c) error...catch
* d) attempt...handle

1. **How do you define middleware in Express.js?**

* a) app.use()
* b) app.middleware()
* c) app.next()
* d) app.route()

1. **Which of the following is a built-in middleware function in Express.js?**

* a) morgan
* b) cookie-parser
* c) express.Router
* d) express.json

**Fill the Answer of Q1 Part a. in this table.**

|  |  |  |
| --- | --- | --- |
| 1. a) style | 15. a) Via props | 29. c) Material-UI |
| 2. a) <h1> | 16. a) this.setState | 30. a) npx create-react-app my-app |
| 3. a) id | 17. a) To help identify which items have changed, are added, or removed | 31. a) PUT |
| 4. a) Ordered List | 18. a) npm install -g package-name | 32. a) res.json() |
| 5. a) background-color | 19. a) 3000 | 33. d) db.collection.copy() |
| 6. a) #main | 20. a) Using the http module | 34. a) npm init |
| 7. a) font-size | 21. a) find() | 35. d) Character |
| 8. a) font-family | 22. a) Create, Read, Update, Delete | 36. d) unshift() |
| 9. a) .container | 23. a) var app = express() | 37. d) Both a and c |
| 10. a) Using rows and columns | 24. a) app.get() | 38. a) try...catch |
| 11. a) function myFunction() {} | 25. a) app.listen() | 39. a) app.use() |
| 12. a) alert("Hello World"); | 26. a) body-parser | 40. d) express.json |
| 13. a) object | 27. a) mongoose.connect() |  |
| 14. a) class MyComponent extends React.Component {} | 28. a) They can be stateful or stateless. |  |

1. **Solve on question paper.**
   1. **Show the output after clicking ShowCount button. If there is any error do not write the output only explain the error. [5 Marks]**

|  |  |
| --- | --- |
| import React from 'react';  class MyComponent extends React.Component {  state = {  count: 0  }  foo(e) {  this.setState({ count: this.state.count -5 });  }  render() {  return (  <div>  <p>Count: {this.state.count}</p>  <button onClick={this.foo}>ShowCount</button>  </div>  );  }  }  export default MyComponent; | Output  No Output |
| Error  foo must be an arrow function in order to use this |

* 1. **After clicking 3 times Add Item button what will be shown on the screen [5 Marks]**

|  |  |
| --- | --- |
| import React from 'react';  class List extends React.Component {  state = {  items: ['Item 1', 'Item 2']  }  addItem= (e) => {  this.setState({  items: ['New Item'] });  }  render() {  return (  <div>  <ul>  {this.state.items.map((item, index) => (  <li key={index}>{item}</li>  ))}  </ul>  <button onClick={this.addItem}>Add Item</button>  </div>  );  }  }  export default List; | Output  New Item |
|  |

* 1. **Write the output. [5 Marks]**

|  |  |
| --- | --- |
| const x = 1  const y = 2  const z = 3  console.count(  'z: ' + z + ' has been checked .. how many times?'  )  setTimeout(function(){  console.count(  'z: ' + z + ' has been checked .. how many times?'  )  }, 1000);  console.count(  'y: ' + y + ' has been checked .. how many times?'  ) | Output  z: 3 has been checked .. how many times?: 1  y: 2 has been checked .. how many times?: 1  z: 3 has been checked .. how many times?: 2 |
|  |

* 1. **What will be the output of the following Express.js script when a GET request is made to /about? [5 Marks]**

|  |  |
| --- | --- |
| const express = require('express');  const app = express();  const port = 3000;  app.use((req, res, next) => {  console.log('t1');  next();  });  app.get('/about1', (req, res) => {  console.log('about1');  });  app.use(function(req, res, next){  console.log('t2');  });  app.post('/about1', (req, res) => {  console.log('about2');  });  app.post('/about', (req, res) => {  console.log('about3');  });  app.get('/about', (req, res) => {  console.log('about4');  });  app.listen(port, () => {  console.log(`Server running at http://localhost:${port}/`);  }); | Output  t1  t2 |
|  |

**Q2: Solve on Answer book [40 marks]**

* + 1. **Create a React component that fetches data from an API endpoint and displays the data in a list. Fetch the data when the component mounts. {API url is** [**www.webapi.com/users**](http://www.webapi.com/users)**, each user object has name, email, and age.} [15 Marks]**

**import React, { Component } from 'react';**

**import axios from 'axios';**

**class UserList extends Component {**

**constructor(props) {**

**super(props);**

**this.state = {**

**users: [],**

**loading: true,**

**error: null,**

**};**

**}**

**componentDidMount() {**

**this.fetchUsers();**

**}**

**fetchUsers = async () => {**

**try {**

**const response = await axios.get('https://www.webapi.com/users');**

**this.setState({ users: response.data, loading: false });**

**} catch (error) {**

**this.setState({ error, loading: false });**

**}**

**};**

**render() {**

**const { users, loading, error } = this.state;**

**if (loading) {**

**return <div>Loading...</div>;**

**}**

**if (error) {**

**return <div>Error: {error.message}</div>;**

**}**

**return (**

**<div>**

**<h1>User List</h1>**

**<ul>**

**{users.map((user) => (**

**<li key={user.email}>**

**<p>Name: {user.name}</p>**

**<p>Email: {user.email}</p>**

**<p>Age: {user.age}</p>**

**</li>**

**))}**

**</ul>**

**</div>**

**);**

**}**

**}**

**export default UserList;**

* + 1. **Consider the question Q2 part a. and instead of third party API create your own web server with express and save the required info in json format and write a route /getUsers with get method. You are required to write a server code and just calling and receiving logic on client side. [10 Marks]**

**// server.js**

**const express = require('express');**

**const app = express();**

**const PORT = 3001;**

**// In-memory JSON data**

**const users = [**

**{ name: "John Doe", email: "john@example.com", age: 30 },**

**{ name: "Jane Smith", email: "jane@example.com", age: 25 },**

**{ name: "Alice Johnson", email: "alice@example.com", age: 35 }**

**];**

**app.get('/getUsers', (req, res) => {**

**res.setHeader('Content-Type', 'application/json');**

**res.send(JSON.stringify(users));**

**});**

**app.listen(PORT, () => {**

**console.log(`Server is running on http://localhost:${PORT}`);**

**});**

**// src/UserList.js**

**import React, { Component } from 'react';**

**import axios from 'axios';**

**class UserList extends Component {**

**constructor(props) {**

**super(props);**

**this.state = {**

**users: [],**

**loading: true,**

**error: null,**

**};**

**}**

**componentDidMount() {**

**this.fetchUsers();**

**}**

**fetchUsers = async () => {**

**try {**

**const response = await axios.get('http://localhost:3001/getUsers');**

**this.setState({ users: response.data, loading: false });**

**} catch (error) {**

**this.setState({ error, loading: false });**

**}**

**};**

**render() {**

**const { users, loading, error } = this.state;**

**if (loading) {**

**return <div>Loading...</div>;**

**}**

**if (error) {**

**return <div>Error: {error.message}</div>;**

**}**

**return (**

**<div>**

**<h1>User List</h1>**

**<ul>**

**{users.map((user) => (**

**<li key={user.email}>**

**<p>Name: {user.name}</p>**

**<p>Email: {user.email}</p>**

**<p>Age: {user.age}</p>**

**</li>**

**))}**

**</ul>**

**</div>**

**);**

**}**

**}**

**export default UserList;**

* + 1. **Consider the Q2 part b. and instead of sending information from json, Get the data from MongoDB. Consider the MongoDB has already contains the collection of users. You are required to write a code of accessing data from MongoDB and sending to the client only. [15 Marks]**

**// server.js**

**const express = require('express');**

**const mongoose = require('mongoose');**

**const app = express();**

**const PORT = 3001;**

**// MongoDB connection URI**

**const mongoURI = 'mongodb://localhost:27017/mydatabase';**

**// Connect to MongoDB**

**mongoose.connect(mongoURI, { useNewUrlParser: true, useUnifiedTopology: true })**

**.then(() => console.log('MongoDB connected'))**

**.catch(err => console.log(err));**

**// Define a user schema and model**

**const userSchema = new mongoose.Schema({**

**name: String,**

**email: String,**

**age: Number**

**});**

**const User = mongoose.model('User', userSchema);**

**// Route to get users**

**app.get('/getUsers', async (req, res) => {**

**try {**

**const users = await User.find({});**

**res.json(users);**

**} catch (err) {**

**res.status(500).json({ error: err.message });**

**}**

**});**

**app.listen(PORT, () => {**

**console.log(`Server is running on http://localhost:${PORT}`);**

**});**

**Q3: Solve on Answer Book [55 marks]**

* + 1. **Consider the Assignment 3 question on Redux. Explain the Redux concept using practical example code, which you have shown in Assignment#3. Note: Assignment marks will be calculated based on this question obtained marks. [10 Marks]**

1. **Store**: The centralized place where the state of the application is stored.
2. **Actions**: Plain JavaScript objects that describe what happened.
3. **Reducers**: Functions that determine how the state should change in response to an action.
4. **Dispatch**: A method used to send actions to the store.
5. **Connect**: A function provided by react-redux to connect React components to the Redux store.

### Practical Example: Managing a List of Books

#### Step 1: Install Dependencies

First, install redux and react-redux:

sh

Copy code

npm install redux react-redux

#### Step 2: Set Up Redux Store

**src/store.js**

javascript

Copy code

import { createStore } from 'redux';

import rootReducer from './reducers';

const store = createStore(

rootReducer,

window.\_\_REDUX\_DEVTOOLS\_EXTENSION\_\_ && window.\_\_REDUX\_DEVTOOLS\_EXTENSION\_\_()

);

export default store;

#### Step 3: Define Actions

**src/actions/bookActions.js**

javascript

Copy code

export const ADD\_BOOK = 'ADD\_BOOK';

export const REMOVE\_BOOK = 'REMOVE\_BOOK';

export const addBook = (book) => ({

type: ADD\_BOOK,

payload: book

});

export const removeBook = (isbn) => ({

type: REMOVE\_BOOK,

payload: isbn

});

#### Step 4: Create Reducers

**src/reducers/bookReducer.js**

javascript

Copy code

import { ADD\_BOOK, REMOVE\_BOOK } from '../actions/bookActions';

const initialState = {

books: []

};

const bookReducer = (state = initialState, action) => {

switch (action.type) {

case ADD\_BOOK:

return {

...state,

books: [...state.books, action.payload]

};

case REMOVE\_BOOK:

return {

...state,

books: state.books.filter(book => book.isbn !== action.payload)

};

default:

return state;

}

};

export default bookReducer;

**src/reducers/index.js**

javascript

Copy code

import { combineReducers } from 'redux';

import bookReducer from './bookReducer';

const rootReducer = combineReducers({

bookList: bookReducer

});

export default rootReducer;

#### Step 5: Set Up Provider in React

**src/index.js**

javascript

Copy code

import React from 'react';

import ReactDOM from 'react-dom';

import { Provider } from 'react-redux';

import store from './store';

import App from './App';

ReactDOM.render(

<Provider store={store}>

<App />

</Provider>,

document.getElementById('root')

);

#### Step 6: Create React Components

**src/components/BookList.js**

javascript

Copy code

import React from 'react';

import { connect } from 'react-redux';

import { removeBook } from '../actions/bookActions';

const BookList = ({ books, removeBook }) => {

return (

<div>

<h2>Book List</h2>

<ul>

{books.map((book, index) => (

<li key={index}>

{book.title} - {book.isbn}

<button onClick={() => removeBook(book.isbn)}>Remove</button>

</li>

))}

</ul>

</div>

);

};

const mapStateToProps = (state) => ({

books: state.bookList.books

});

const mapDispatchToProps = {

removeBook

};

export default connect(mapStateToProps, mapDispatchToProps)(BookList);

**src/components/AddBook.js**

javascript

Copy code

import React, { useState } from 'react';

import { connect } from 'react-redux';

import { addBook } from '../actions/bookActions';

const AddBook = ({ addBook }) => {

const [title, setTitle] = useState('');

const [isbn, setIsbn] = useState('');

const handleSubmit = (e) => {

e.preventDefault();

addBook({ title, isbn });

setTitle('');

setIsbn('');

};

return (

<form onSubmit={handleSubmit}>

<div>

<label>Title:</label>

<input

type="text"

value={title}

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

/>

</div>

<div>

<label>ISBN:</label>

<input

type="text"

value={isbn}

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

/>

</div>

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

</form>

);

};

const mapDispatchToProps = {

addBook

};

export default connect(null, mapDispatchToProps)(AddBook);

**src/App.js**

javascript

Copy code

import React from 'react';

import BookList from './components/BookList';

import AddBook from './components/AddBook';

function App() {

return (

<div className="App">

<h1>Book Manager</h1>

<AddBook />

<BookList />

</div>

);

}

export default App;

* + 1. **In the Express Server topic we have studied about Templates. You are required to use the templates for showing books data on requesting /showBooks route. The book object contains isbn, title, and price. [15 Marks]**

**App.js**

**const express = require('express');**

**const app = express();**

**const port = 3000;**

**// Set EJS as the templating engine**

**app.set('view engine', 'ejs');**

**// Sample book data**

**const books = [**

**{ isbn: '978-3-16-148410-0', title: 'Book One', price: 29.99 },**

**{ isbn: '978-1-4028-9462-6', title: 'Book Two', price: 39.99 },**

**{ isbn: '978-0-12-374856-0', title: 'Book Three', price: 24.99 }**

**];**

**// Route to display books**

**app.get('/showBooks', (req, res) => {**

**res.render('books', { books: books });**

**});**

**// Start the server**

**app.listen(port, () => {**

**console.log(`Server is running on http://localhost:${port}`);**

**});**

**views/books.ejs**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<title>Books List</title>**

**<style>**

**table {**

**width: 100%;**

**border-collapse: collapse;**

**}**

**table, th, td {**

**border: 1px solid black;**

**}**

**th, td {**

**padding: 15px;**

**text-align: left;**

**}**

**th {**

**background-color: #f2f2f2;**

**}**

**</style>**

**</head>**

**<body>**

**<h1>Books List</h1>**

**<table>**

**<tr>**

**<th>ISBN</th>**

**<th>Title</th>**

**<th>Price</th>**

**</tr>**

**<% books.forEach(function(book) { %>**

**<tr>**

**<td><%= book.isbn %></td>**

**<td><%= book.title %></td>**

**<td><%= book.price %></td>**

**</tr>**

**<% }); %>**

**</table>**

**</body>**

**</html>**

* + 1. **Explain the following routes and provide the examples of urls which can be called on the following routes: [7\*4=28 Marks]**

1.

app.get('/ab?cd', (req, res) => {

  res.send('ab?cd')

})

* b zero time or 1 time
* This route path will match acd and abcd.

2.

app.get('/ab+cd', (req, res) => {

  res.send('ab+cd')

})

* b 1 time or many time
* This route path will match abcd, abbcd, abbbcd, and so on.

3.

app.get('/ab\*cd', (req, res) => {

  res.send('ab\*cd')

})

* Anything 0 or many times
* This route path will match abcd, abxcd, abRANDOMcd, ab123cd, and so on.

4.

app.get('/ab(cd)?e', (req, res) => {

  res.send('ab(cd)?e')

})

* (cd) 0 or many time
* This route path will match /abe and /abcde.

5.

app.get(/.\*fly$/, (req, res) => {

  res.send('/.\*fly$/')

})

* Anything in the start but end at fly
* This route path will match butterfly and dragonfly, but not butterflyman, dragonflyman, and so on.

6.

app.get('/flights/:from-:to', (req, res) => {

  res.send(req.params)

})

* :from and :to can be replace with any value, - will be used hardcoded
* flights/LAX-SFO

7.

app.all('/secret', (req, res, next) => {

  console.log('Accessing the secret section ...')

  next()

})

* all methods i.e. get, post, put on secret route will land here
  + 1. **Bonus: Draw a smiley. Keep smiling and brighten someone's day! [2 Marks]**