Exam - Web Development TX00EY23-3001

- Part 1: React (worth 150 points)
- Part 2: Backend (worth 200 points)
- Part 3: *Bonus* (30 points)

Part 1

React: Question A (45 points)

1. How does React integrate into the MERN stack?
2. Describe the notion of component composition and its benefits in React application development.
3. Analyze the following example and provide an explanation of the purpose of the *useEffect* hook and its appropriate use cases.

```
function Timer() {
  const [count, setCount] = useState(0);

  useEffect(() => {
    let timer = setTimeout(() => {
      setCount((count) => count + 1);
    }, 1000);

    return () => clearTimeout(timer);
  }, []);

  return <h1>The component has rendered {count} times!</h1>;
}
```

4. Describe how this component applies conditional styling based on the *type* prop.

```
function Alert({ type }) {
    const styles = {
        success: { color: 'green' },
        error: { color: 'red' },
    };

    return <div style={styles[type]}>This is a {type} alert.</div>;
}
```

React: Question B (25 points)

Here's the logic for a custom hook designed for a generic input field.

1. What does this hook return?
2. Can we name this hook *myHook*? If not, justify your answer and provide an alternative name(s) for the hook.
3. What happens if, instead of writing **import { useState } from "react"**, we write **import useState from "react"**?

```
import { useState } from "react";

const myHook = (type) => {
  const [value, setValue] = useState("");
  const onChange = (event) => {
    setValue(event.target.value);
  };
  return { type, value, onChange };
};
```

React: Question C (80 points)

The **Signup** component below is for user registration. Refer to the code and:

1. Extract the logic for *signing up* into a custom hook.
2. How can you use the extracted hook in the *Signup* component?
3. Describe the mechanism for displaying an error message when the **error** state is **true**.
4. Modify the button within the **Signup** component so that it is disabled when **isLoading** state is **true**.
5. Explain the purpose of the following line: **body: JSON.stringify({ email, password })**.
6. Justify the difference between using **setIsLoading(true)** and **isLoading=true** when managing the loading state.
7. Discuss the significance of using **e.preventDefault()** in the *handleSubmit* function and what might happen if it were omitted.
8. Explain the difference between using **onChange={(e) => setEmail(e.target.value)}** and **onChange={setEmail(e.target.value)}** when handling input changes.
9. What is the difference between <form className= "signup" onSubmit={handleSubmit}> and <form class= "signup" onsubmit=handleSubmit()>.

```
import { useState } from "react"
import { useAuthContext } from './useAuthContext'
const Signup = () => {
  const [email, setEmail] = useState('')
  const [password, setPassword] = useState('')
  const [error, setError] = useState(null)
  const [isLoading, setIsLoading] = useState(false)
  const { dispatch } = useAuthContext()
```

```
  const signup = async (email, password) => {
    setIsLoading(true)
    setError(null)

    const response = await fetch('/api/user/signup', {
      method: 'POST',
      headers: {'Content-Type': 'application/json'},
      body: JSON.stringify({ email, password })
    })
    const data = await response.json()

    if (!response.ok) {
      setIsLoading(false)
      setError(data.error)
    }
    if (response.ok) {
      localStorage.setItem('user', JSON.stringify(data))
      dispatch({type: 'LOGIN', payload: data})
      setIsLoading(false)
    }
  }

  const handleSubmit = async (e) => {
    e.preventDefault()
    await signup(email, password)
  }

  return (
    <form className="signup" onSubmit={handleSubmit}>
      <h3>Sign Up</h3>
      <label>Email address:</label>
      <input
        type="email"
        onChange={(e) => setEmail(e.target.value)}
        value={email}
      />
      <label>Password:</label>
      <input
        type="password"
        onChange={(e) => setPassword(e.target.value)}
        value={password}
      />
      <button>Sign up</button>
    </form>
  )
}

export default Signup
```

Part 2

Backend: Question A (70 points)

1. How does express fit in the MERN stack?
2. How do you use middleware to handle authentication in an Express.js application?
3. How do controllers help in maintaining separation of concerns in the MVC pattern?
4. Describe the difference between authentication and authorization.
5. How does token-based authentication work, and why is it commonly used in web applications?
6. Describe the advantages of using **async/await** over promise chaining with **.then()** for handling asynchronous operations.
7. What testing tools and libraries are commonly used for testing Express.js applications?
8. Compare the process of testing an API server with **Postman** and **Supertest**.

Backend: Question B (65 points)

Refer to the following snippet:

1. To which HTTP method, **ACTION1**, **ACTION2** and **ACTION3** belong? Justify?
2. Explain this line of code: **books[bookIndex] = { id: bookId, ...updatedBook}**

```
app.ACTION1("/api/books/:id", (req, res) => {
  const bookId = parseInt(req.params.id);

  const bookIndex = books.findIndex((book) => book.id === bookId);

  if (bookIndex !== -1) {
    books.splice(bookIndex, 1);
    res.json({ message: "Book message 1" });
  } else {
    res.status(404).json({ message: "Book message 2" });
  }
});
```

```
app.ACTION2("/api/books", (req, res) => {
  const book = {
    id: books.length + 1,
    title: req.body.title,
    author: req.body.author,
  };

  books.push(book);
  res.status(201).json(book);
});
```

```
app.ACTION3("/api/books/:id", (req, res) => {
  const bookId = parseInt(req.params.id);
  const book = req.body;

  const bookIndex = books.findIndex((book) => book.id === bookId);

  if (bookIndex !== -1) {

    books[bookIndex] = { id: bookId, ...book };
    res.json(books[bookIndex]);
  } else {
    res.status(404).json({ message: "Book message" });
  }
});
```

Backend: Question C (65 points)

1. Is there anything wrong with this code? if so how can you fix it?

```
const app = express();
app.use(cors());

app.use(errorHandler);
app.use(express.urlencoded({ extended: false }));

app.use("/api/goals", goalRoutes);
app.use("/api/users", userRoutes);

app.get("/", (req, res) => res.send("Hello"));
app.use(express.json());

app.listen(port, () => console.log(`Server started on port ${port}`));
```

2. Explain the purpose of this middleware when used with Express.js routers.

```
app.use((err, req, res, next) => {
// .
});
```

3. What are the potential consequences if **node_modules/** and **.env** are not included in the **.gitignore** file?

4. Refer to the snippet below extracted from **package.json** file:

- How can you execute the **start:backend** script?
- Why are **jest** and **express** located under different categories in the `dependencies`?

```
{
  "scripts": {
       "start:backend": "cross-env NODE_ENV=production node backend/index.js",
       "dev": "cross-env NODE_ENV=development node backend/index.js"
  },
  "devDependencies": {
       "cross-env": "^7.0.3",
       "jest": "^29.7.0"
  },
  "dependencies": {
       "dotenv": "^16.3.1",
       "express": "^4.18.2"
  }
}
```

5. Refer to the snippet below extracted from **.env** (written by one of your classmates):

- Explain what he meant by this line **JWT_SECRET = 64bytesofrandomness**. *Please provide a detailed response, NOT a superficial answer.*
- Explain how you can access the **MONGO_URI** environment variable, from a node module.

```
JWT_SECRET = 64bytesofrandomness
MONGO_URI=mongodb//usr:pssw@cluster.mongodb.net
```

Part 3: Bonus

> To be eligible for a bonus through self-assessment, you need to be fair in your judgment.

1. On a scale of 1 to 5, evaluate your performance in **this exam**. (**5 points**)
2. Using a scale of 1 to 5, assess your performance in the **web development course**. (**5 points**)
3. Using a scale of 1 to 5, assess your performance in the **project course**. (**5 points**)
4. What **qualities** (characteristics) define a full stack developer? (**10 points**)
5. Define **soft skills** and elaborate on their importance within a work environment. (**5 points**)