CFGDEGREE

FULLSTACK ASSESSMENT MATERIAL RELEASE

THEORY QUESTIONS

SECTION TYPE	TOTAL MARKS AVAILABLE NOTES
Design heuristics	10
Redux	10
React	10
Agile	10
40 marks available total	

Important notes:

- This document shares the first section of the FullStack Assessment which is composed of 4 FullStack Theory Questions
- You have 24 hours before the assessment to prepare.
- If any plagiarism is found in how you choose to answer a question you will receive a 0 and the instance will be recorded. Consequences will occur if this is a repeated offence. You can remind yourself of the plagiarism policy here.
- Answers need to be explained clearly and illustrated with relevant examples where necessary. Your examples can include code snippets, diagrams or any other evidence-based representation of

your answer.

`Questions begin on the next page

Theory Questions	**40 total**

1. In design Heuristics, what does the term “advantages of Matching between system and the real world” mean? What are the advantages?

ANS:

The term "match between system and the real world" refers to a usability heuristic that states that a system should be designed in a way that is consistent with the real world. This means using familiar language, terminology, and concepts, and following real-world conventions. For example, when you design a game and you include a lobby, users know this is an entrance that leads to multiple possibilities, i.e, leaderboard, game levels, et cetera. In other words, the design should mirror the user's expectations based on their familiarity with how things work in the real world.

Advantages include:

Familiarity - Users find it easier to interact with a system when its elements and interactions resemble familiar real-world objects and actions. This reduces the cognitive load, and time required to understand the system.

User Satisfaction - When users find a system easy to understand and use, they are more likely to feel satisfied with their experience, leading to positive perceptions and potential repeat usage.

Accessibility - Designs that match the real world are often more accessible to a wider range of users, as they possess tools that are able to support people. For example, those who may not be familiar with complex digital interfaces, disabled and/or neurodiverse individuals.

> Reduced Training   Systems that match the real world require less extensive training for users to become proficient, as the learning curve is smoother.
>
> Reduction in Documentation - If the design is aligned with the real world, the need for extensive documentation or tutorials might be reduced, as users can rely on their existing knowledge.

2. What do you understand by “Single source of truth”? and how does it relate to redux? What are the advantages ?

ANS:

A single source of truth (SSOT) is a principle in software engineering and data management that refers to the practice of aggregating the data from many systems within an organisation to a single location. A SSOT is not a system, tool, or strategy, but rather a state of being for a company’s data in that it can all be found via a single reference point (MuleSoft, 2023) .This ensures that everyone in the organization is working with the same data, which can help to prevent errors and inconsistencies.

Redux is a state management library for JavaScript applications. It follows the SSOT principle by storing the entire state of an application in a single store. This makes it easy to track changes to the state and ensure that everyone in the application is working with the same data.

Here's how "single source of truth" relates to Redux:

Predictable State Changes: Redux enforces a strict pattern for updating the state called "actions" and "reducers." Actions are dispatched to describe what change should occur, and reducers are pure functions that specify how the state should change in response to those actions. This pattern ensures that state changes are predictable and maintainable.

Consistency - Everyone in the application is working with the same data, which can help to prevent errors and inconsistencies. This eliminates the risk of data inconsistency that can arise when different parts of the application hold separate copies of the same data.

Debugging - Having a single source of truth simplifies debugging. Developers can track how the state changes over time by reviewing the sequence of dispatched actions and their corresponding state changes in the store.

Efficient Updates - Redux uses a mechanism called "state diffing" to efficiently update the UI. When the state changes, Redux calculates the difference between the old and new state and updates only the necessary components in the UI.

Time Travel Debugging - Redux's single source of truth makes it possible to implement features like time travel debugging, where developers can replay the application's state changes to identify and fix issues.

Performance - Redux can help to improve the performance of an application by reducing the amount of data that needs to be passed between components.

Scalability - As applications grow in complexity, managing state becomes challenging. Redux's single source of truth makes it easier to manage and reason about the state as the application scales.

Team Collaboration - With a single source of truth, team members can have a shared understanding of the application's data structure and state management, leading to smoother collaboration.

Overall, the SSOT principle, as implemented in Redux, can help to improve the development and maintenance of software applications. Additional advantages also include easy testing, scalability and maintainability.

3. What is the difference between a stateless component and a stateful component in React?

ANS:

The difference between stateful and stateless is that one has state, and the other doesn’t. That means the stateful components are keeping track of changing data, while stateless components print out what is given to them via props, or they always render the same thing.

For example,

Stateless component :

This example uses the tv show, "The Vampire Diaries" (TVD) to explain stateless components. In this instance, a stateless component could display a brief summary of a specific episode. The component only needs the episode information passed as props and doesn't manage its own state

```
const EpisodeSummary = (props) => {

  return (

    <div>

      <h3>{props.title}</h3>

      <p>{props.summary}</p>

    </div>

  );

};
```

State component:

In this instance, a stateful component could represent a character's current emotional state. The component could have buttons to update the character's emotions, and it would manage the state changes accordingly.

```
class EmotionalState extends React.Component {

  constructor(props) {

    super(props);

    this.state = {

      emotions: "Neutral",
    };
  }


  setEmotion = (emotion) => {
    this.setState({
      emotions: emotion,
    });
  };


  render() {
    return (
      <div>
        <h3>{this.props.character}</h3>
        <p>Emotions: {this.state.emotions}</p>
        <button onClick={() => this.setEmotion("Happy")}>Happy</button>
        <button onClick={() => this.setEmotion("Sad")}>Sad</button>
        <button onClick={() => this.setEmotion("Angry")}>Angry</button>
      </div>
    );
  }
}
```

4. List out the advantages and disadvantages of exploratory testing (used in Agile) and scripted testing?

Exploratory Testing	Scripted Testing
Testers need a thorough knowledge of the domain and expertise.	Testers can overcome any lack of knowledge about the domain during the design stage. And with documentation, they have notes to discuss and improve upon.
No formal documentation, only exploratory documentation.	A high level of documentation is mandatory
Little to no lead-in time is needed for test execution.	A significant lead-in time is required.
Exploratory testing emphasizes learning and adaptability.	Scripted testing focuses on decision-making and prediction.
No measurable or transparent test coverage. The test results are based on a tester's report.	Testers can trace the test scripts and documentation to demonstrate test coverage.