

**FULLSTACK ASSESSMENT MATERIAL RELEASE**

(Alternative) THEORY QUESTIONS

|  |  |  |
| --- | --- | --- |
| **SECTION TYPE** | **TOTAL MARKS AVAILABLE** | **NOTES** |
| **Design heuristics** | 10 |  |
| **Agile** | 10 |  |
| **React** | 20 |  |
| **40 marks available total** | | |

**Important notes:**

* This document shares the first section of the Full Stack Exam which is composed of 3 Full Stack Theory Questions
* You have 24 hours before the exam 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](https://drive.google.com/file/d/1k9UaGOR7hx54QRZ8jvp2jtC4P-8_Rs4F/view?usp=sharing).
* 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. Explain why it’s important to be mindful of consistency and standards when designing an app. *Make sure to point out the advantages and if any the disadvantages.* (10 marks)

Answer-

Design consistency is really important in making apps easier and more enjoyable to use. When an app is consistent, users can quickly understand how things work because everything feels familiar and predictable. This not only makes the app look more polished but also helps users avoid mistakes.

Advantages-

Easier to Use: When elements behave in a predictable way, users can navigate the app smoothly without confusion.

Quick to Learn: Once users get the familiar with one part of the app, they can easily figure out the rest without needing much extra help.

Looks Better: A consistent design makes the app visually appealing and professional, which creates a better impression.

Fewer Mistakes: When users know what to expect, they’re less likely to make errors, making the app more reliable.

Stronger Brand Identity: Using the same visual and functional elements throughout the app helps reinforce the brand, making it more recognizable to users.

For example- Microsoft office, the Microsoft office uses consistent design of ribbons and icons for copy, paste, creating new file, save file etc.

Disadvantages-

Lack of Creativity: Focusing too much on consistency can limit creativity and innovation, making designers feel restricted by established patterns.

Rigid User Experience: Strictly following standards might not always suit unique user needs or situations, which can prevent necessary changes.

1. What is Agile testing? What testing is done during Agile? What is the difference between a traditional Waterfall testing and Agile testing? (10 marks)

Answer-

Agile Testing and Waterfall Testing are two different approaches to software testing. Agile testing happens along with development cycle. This type of testing continuously happens during the project development, ensuring any bugs are identified and resolved as quickly as possible.

During the Agile testing different types of testing are done-

Unit Testing: Testing each component of the software separately.

Integration Testing: Testing different components work together

Regression Testing: This is testing, to check system is still working after modification or if new features are added into the system.

Acceptance Testing: Checking that the software meets user requirements, acceptance testing done at the end of each iteration.

Exploratory Testing: Testers explore the application to find issues that automated tests might miss.

**Agile Testing:**

Ongoing Process: Testing happens continuously throughout development, not as a separate phase.

Integrated with Development: Testing is done alongside coding, allowing issues to be identified and fixed quickly.

Team Collaboration: The development and testing teams work closely together, fostering teamwork and communication.

Tester Involvement: Testers are involved from the start, even during the requirements phase, ensuring a better understanding of the project.

Acceptance Testing: After each iteration of development, acceptance testing is performed to make sure the product meets the requirements.

Regression Testing: In Agile testing, regression testing (checking if new code changes have affected existing functionality) happens after each round of development

Time Delay:

In Agile testing, there’s no delay between coding and testing; they happen together.

**Waterfall Testing:**

Distinct Phase: Testing is done after development is fully completed, as a separate phase.

Sequential Process: Testing only begins once the entire development phase is finished, which can delay finding and fixing issues.

Separate Teams: The development and testing teams usually work separately, which may reduce collaboration.

Limited Tester Involvement: Testers might not be involved early in the project, which could lead to gaps in understanding the requirements.

Acceptance Testing: Acceptance testing typically occurs at the end of the project, once all development and initial testing are complete.

Regression Testing: In waterfall, regression testing is done only at the very end of the project.

Time Dealy: In Waterfall testing, there’s usually a time delay between when coding finishes and testing begins.

* 1. What are the limitations of React? (10 marks)

Answer-

As a former web developer, I personally found that React is a bit difficult to learn, and the syntax can be complicated since it combines XML, HTML and JavaScript. React introduces concepts like components, state management, and hooks, which can be challenging for beginners especially new programmers. And, as project grows managing state, components and props can be more complicated.

Also, the documentation isn't always well-explained, and there are often gaps between the introduction of new features in React and the availability of corresponding documentation. These gaps can make it difficult for developers to fully understand and implement new features, especially when they are newly released.

Since React is a library and not a framework, there is no predefined folder or file structure, so developers need to decide on this themselves and this leads to inconsistencies across different teams.

Additionally, React requires an advanced understanding of JavaScript, this includes understanding ES6+ features, which not all developers may be familiar with.

React releases new versions of the library several times a year. Each release includes new features as well as some breaking changes. Developers need to quickly learn and apply these new concepts to keep the application stable.

In addition of all these- React only handles UI part, it needs other tools or packages to handle other parts of the app. For example, React often needs other libraries for routing, state management, etc., which can add more complexity.

* 1. What are Error boundaries in React and How do you handle error handling in a React application (10 marks)

Answer-

An error boundary is a special React component that is used to catch and handle errors in other parts of the application. It acts like a safety net, stopping errors from spreading and causing problems in the rest of your app. This way, even if one part of your app has an error, the rest of it can keep working without issues.

Although React has its own way to handle errors with something called **componentDidCatch**, this only works for class components and not for event handlers. Error boundaries are better because they work with both class and function components. They help make sure that unexpected problems, like bugs or network issues, don't cause entire app to crash.

When an error happens in a component that is wrapped by an error boundary, React calls the **componentDidCatch** method of the error boundary. This method gets two pieces of information: the error itself and some extra details about where the error happened in the component tree.

The **componentDidCatch** method lets you decide what to do with the error. You can log it, show a fallback message to the user, or display a different part of the app to recover from the error smoothly. The error boundary helps control the error and keeps the rest of the app running normally.

For example, you have an application with a component called Dashboard. You can wrap the Dashboard component with an error boundary to catch any errors that might happen inside Dashboard or any of its child components. If something goes wrong within the Dashboard, the error boundary will catch the error and show a fallback UI, like an error message or a special component that handles the error in a user-friendly way.

class ErrorBoundary extends React.Component {

constructor(props) {

super(props);

this.state = { hasError: false };

}

componentDidCatch(error, errorInfo) {

// Log the error or send it to an error reporting service

console.error(error, errorInfo);

// Update state to show the fallback UI

this.setState({ hasError: true });

}

render() {

if (this.state.hasError) {

// You can render any custom fallback UI

return this.props.fallback;

}

return this.props.children;

}

}

export default ErrorBoundary;

// Usage

<ErrorBoundary fallback={<span>Something went wrong</span>}>

<Dashboard/>

</ErrorBoundary>