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

The term refers to using principles that are reflective of real life, such as the use of language,
concepts and behaviours and embedding those within the system that has been created. The impact of this
means that users will be able to operate with a system just as naturally as they would within the real
world which most of the time is automatic, meaning that the users experience will be a smoother and
easier one by adopting this principle.

The advantages of adopting this principle within practice are endless. We can ensure that there is
minimal learning to undertake from the user as they are working with concepts that they use in the
real world, an example of this would be using a website that uses an icon of a trolley to represent
all of the products they have selected, the lack of learning needed increases accessibility and speed
in which a user can navigate a system, with a more intuitive use.

The familiarity of a system will mean that the user is more likely to enjoy the process of using the
system which is reflective of their real world, and thus more likely to leave positive reviews and
have an overall positive user experience.

With this principle we might expect to also see fewer errors being made by the user, if we use another
example of a website that uses folders and to represent these they are in the shape of a tangible
folder, they are able to make quicker connections on how to use the system and what it means.

2. What do you understand by "single source of truth"? How does it relate to redux? What are the
advantages?

The single source of the truth in regards to Redux refers to creating a single centralised store which
holds all the state of your entire application. The store is an object that holds the state tree for
the whole project, so all attributes.

Advantages of having a single source of the truth within Redux, means that it is easier to manage
larger applications, with all of the state logic being in one centralised place, so you are able to
handle state changes and transformations with more ease. With this ease it also means that when having
multiple developers working on an application they will be aware that all of the state logic will be
in one place, making it easier for multiple people to work on it and develop it further.

With the single source of the truth, you are ensuring consistency within your application, with all
components being able to access the same state, it is not only easier to manage but also easier when
it comes to debugging allowing the developer to track changes easily, with tools like Redux DevTools
which are optimised for a single state tree.

Further advantages will see that if any rules would need to change that would impact the application
updates would be easy to implement, with the logic all being in one place. Finally an advantage to
single source of truth in Redux is the community that works with and contributes to it, with lots of
resources available in order to allow other to understand them more and create more reliable
applications with the focus on a centralised state.

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

A stateless component, receives data through props and renders the user interface based on the
properties which it is using. Once the data is rendered it doesn't remember this data or manage
any changes to it. Below is an example of a stateless component using props to receive data.

```
<!-- const UserBio = (props) => {
  return (
    <div>
      <h2>{props.name}</h2>
      <p>Age: {props.age}</p>
      <p>Number: {props.number}</p>
    </div>
  );
};
```

```
<UserBio name="Bob Ross" age={60} number={07751 678910}/> -->

In contrast to the stateless component, the stateful component can manage its
own state. The state is
an object which holds information and attributes that can be manipulated and
changed over the lifecycle.

<!--import React, { useState } from 'react';

const Counter = () => {
  const [count, setCount] = useState(0);

  const increment = () => {
    setCount(count + 1);
  };

  const decrement = () => {
    setCount(count - 1);
  };

  return (
    <div>
      <h1>Counter {count}</h1>
      <button onClick={increment}>Increment </button>
      <button onClick={decrement}>Decrement </button>
    </div>
  );
};

export default Counter; -->

The stateful component can retain data and updates the user interface in
accordance to this, because of
this stateful components tend to be able to handle more complexities due to
the state management. However
the stateless components might see a stronger performance as they do not
manage the state, and thus not
having to perform as many re-renders.

Therefore a stateless component would be best suited to a more simple
application whereas a stateful
component can handle stateful date with more dynamic interactions.

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

Exploratory testing is set to engage the tester more within the software and the tester creates tests on
a more ad-hoc basis based on what they have learnt, experienced or come across whilst engaging with the
software. The advantages of this type of testing allows for adaptability responding to change which is
a key part of the Agile way of working. Exploratory testing also encourages diversity and creativity from
the tester, where they can implement techniques that might not be anticipated within scripted testing.

Although there are advantages to exploratory testing, the disadvantages that could be presented would be
having a greater dependence on the skill level and creativity of the developer working on those tests
which might not be as impactful if you have a developer with a lack of experience. It can also pose as
tricky to track the progress of exploratory tests due to the ad-hoc nature of them it means there is no
standard checklist. Finally it can also mean that certain tests might be missed or overlooked with this
way of working, which could mean bugs being missed.

Scripted testing is quite different, with scripted testing we see detailed test cases that are predefined
before any testing starts. A tester will have everything ready before hand to execute any tests such as
the instructions, predicted results, and the conditions for the tests to be run under. An advantage to this
way of testing will mean that this will be great for automation, with the consistency that is shown within
the preparation to these tests, so these tests can be run the same way every time.

With details of the tests being predefined before the tests are written it means that the documentation
for these tests will be detailed and reliable, which is useful for other developers wanting to look at
this documentation for referencing of audit.

The disadvantages that a developer might run into with scripted testing would be the time management of
these tests, there is a lot of preparation that has to go into these tests before they are even run which
could pose an issue when you have tight deadlines. Unlike the flexibility for exploratory testing, scripted
testing sees an inflexible system which might not be as adaptable to change, missing those unexpected

```
bugs that might be unearthed within exploratory testing.
```