1. **We're looking for people with a real passion for collaboratively creating great software. Please give an example of a software component you have designed and written from concept to deployment, outlining the steps you took.**
    - While working at National Express, I noticed during the booking engine (the process from selecting a ticket - to purchasing) Customers would drop out during the final stages of the process (using analytic software to gather customer information) .
    - The hypothesis; The Booking summary component being too large on mobile devices and therefore restricted their view
    - I made a responsive web component using JavaScript , CSS and HTML on the last stage of the booking engine. The component used touch event listeners to dynamically display and hide, based on scroll position.
      The component was first built as an AB test, once successful. I built it with the development team using React and Angular frameworks.
    - This process was repeated multiple times for the contact us page, Journey Planner and various pages on the website. (Analyse data, Hypothesis, Test, Build , Deploy)
2. **Using the example that you provided above, tell us about a significant decision you made to solve a technical challenge. Give details of technologies that you chose and why you chose them.**
    - When working with the digital agency 4-roads, I was tasked with designing and building a Hitachi server rack website. One of the most challenging aspects was dealing with state. The team was previously only exposed to Redux. However for a smaller App, that didn't need the extra files. I used the Context API.
    - The Context API is native to React, it requires less code. It's used to remember the global 'State' of the app.
      Another challenge was the design, I used a custom css layout using flex box and Bootstrap.

3. **Using the example that you provided above, tell us about how you ensured your software was fit for purpose and of high quality. What did you learn and what would you do differently next time to do a better job?**
    - Software development inevitably involves bugs, to reduce the app breaking ones; I use JS docs and coding comments to ensure the parameters and variables of functions are clearly understood.
    - Documentation in code is also very important, Making sure functions have one purpose.
    - Variable names should be descriptive, avoid anonymous functions (harder to debug), instead give them a name. Don't cloud the globals, use encapsulation. Make such functions have little to no side effects (unless that's the intended purpose). Perform garbage collection of objects not in use.
    - Code should be readable, so in the future you can understand it along with the team. Using design patterns helps. I used BEM for the above example

- Not only this, But on bigger team projects; I will use TypeScript. TypeScript is strictly typed, meaning you must state the data types of variables and use abstractions
- I also use es-lint and testing software such as Jest to ensure my code has coverage. Having a second pair of eyes for the coding review also helps with readability and maintainability.
- I use PR reviews as a learning lesson. I can get valuable information for how to write better in the future and learn more about my mistakes so I don't make them again in the future.