# CSE 300 – Week 04 Assignment Report

Name:

## Informational Interview Progress

During this semester you need to complete three informational interviews. Part of this week’s assignment is to report on your progress.

1. How many of the three informational interviews have you completed so far this semester?

0

1. If you have not completed three informational interviews yet, you should contact at least five people each week to try to arrange these. List the people you contacted this week for potential informational interviews:

|  |  |  |
| --- | --- | --- |
|  | Person | Company |
| 1 | Steve Mustanski | Capital One |
| 2 | Shane Carleton | MapLarge |
| 3 | Chris Anderson | Centennial Technologies |
| 4 | Caique Ribeiro Rodrigues | Avantos.ai |
| 5 | Joey Lamborn | Acorns |

## Reflection Questions:

1. What was the most important insight you gained from this assignment?

I think the biggest insight I gained was that thinking of code on the spot is harder than I remember, and I need to focus more on formatting for the languages I plan to interview for so that I don’t mix them up during an interview.

2. What was the most difficult question you encountered?

The most difficult questions were those that were asking about technologies I’m not as familiar with, like when I was asked about React or Mongo, so I probably need to study up on those before interviews.

3. What are three things that you can improve on in your technical interviewing?

Explaining my processes, researching beforehand, and fully understanding what is being asked.

## Interview #1 Transcript

(Copy and paste the transcript of your AI interview here)

<https://chatgpt.com/share/68293379-c464-8000-b2e1-0c9a431e9f0f>

## Interview #2 Transcript

(Copy and paste the transcript of your AI interview here)

Take on the role of a technical job interviewer for a software development job, who is interviewing a new college graduate for a software development position. Ask me questions like you would in a technical interview.

Alright, welcome! Thanks for coming in today. I'm excited to chat with you and learn more about your background and skills.

To start, could you walk me through a recent project you worked on, perhaps a significant one from your coursework or a personal endeavor? Please describe the problem you were trying to solve, the technologies you used, and the challenges you faced along the way.

I was recently working on an app that tracks macros and compares them with a users weight to see trends. We used a nutrition api that was difficult to work with as it needed a backend to get around CORS, and needed a consisten IP.

That sounds like an interesting and practical project! Dealing with CORS issues and the need for a consistent IP address when working with external APIs is a common challenge.

Could you elaborate a bit more on how you addressed the CORS issue and the consistent IP requirement? What specific technologies or approaches did you implement on the backend to act as an intermediary for the nutrition API?

I used nodeJS and express as a proxy to help get around cors, and found a webhost VMI that allowed to to have access to a stable IP.

Great! Using Node.js and Express.js to create a proxy is a solid approach for handling CORS. It gives you a lot of control over the requests and responses.

Can you tell me a bit more about the structure of your Express.js proxy? How did you configure it to forward requests to the nutrition API, and how did you handle the API responses before sending them back to your front-end application? Were there any specific middleware or techniques you used for this?

The express proxy would listen for a request from the client, then it would get the data and use it to send a fetch request to the api, then after it received a response, it would check if there was data, and send that data back to the client.

Okay, that's a good high-level overview. Let's dive a little deeper into the data handling.

When your Express.js proxy received the data from the nutrition API, what format was that data typically in (e.g., JSON, XML)? And what steps did you take to ensure the data sent back to your client application was in a format that your front-end could easily consume? Did you perform any data transformation or filtering within your proxy?

We didn't use any filtering, as the data was normalized from the api in a JSON.

That's convenient that the API provided the data in a normalized JSON format. It certainly simplifies the data handling on your proxy.

Now, let's shift gears a bit. Considering the user's weight tracking aspect of your application, how did you structure your database to store and associate the macro data with individual users and their weight entries? What considerations did you have regarding data relationships and efficient querying for trends over time?

It was actually a project where we only were meant to have to use frontend, so other than the api we didn't really use the backend. For storing weight and macros we used localstorage and would simply display a graph of recent macro and weight information based on the dates provided by the user.

Ah, I see. So the focus was primarily on the front-end implementation and utilizing the nutrition API. That clarifies things.

Given that you were working with local storage for persisting the user's weight and macro data, what were some of the trade-offs you considered when choosing this approach, especially in the context of potential scalability or data security? Were there any limitations you encountered or foresaw with using local storage for this type of application?

The biggest problem we saw was that the user would have to use the same device when tracking their information, and that we woudln

You're right, device dependency is a significant limitation of using local storage. If a user switches devices, their data won't automatically transfer. What other limitations or potential issues did you consider or foresee with relying solely on local storage for this kind of data? Think about aspects like data security, data integrity, or the potential for data loss.

sorry, i sent that too early, we also weren't able to use tables and link information together, and were limited to storing information in JSON formats

Thanks for clarifying. The inability to establish relationships between different pieces of data, like you would with tables in a database, is a crucial limitation. You're also spot on about being restricted to storing information primarily in JSON format within local storage. This can make complex data structures and querying more challenging.

Considering these limitations, if you were to expand this application in the future and needed to support multiple devices and more complex data relationships, what kind of database system do you think would be a good fit, and why? What are some of the key features you would look for in a database for this kind of application?

We would probably go for a NoSQL language like mongoDB, as we don't need data to be too structured. We would simply have the weights stored with date information attached, as well as the nutrition information, we would then be able to request the data we wanted to see rather than just checking if data was there with a specific name

That's a reasonable consideration. MongoDB, as a NoSQL database, offers flexibility in terms of schema, which can be advantageous when dealing with evolving data structures. The ability to embed related information directly within documents, like associating weight entries with their dates, can also simplify some querying scenarios.

However, even in a NoSQL database like MongoDB, data modeling is still a crucial aspect. While you have more flexibility than with a strictly relational database, you'd still need to carefully design your collections and documents to ensure efficient querying and data integrity as your application scales and potentially requires more complex relationships in the future (e.g., associating specific meals with macro breakdowns).

Could you elaborate on how you envision structuring your MongoDB documents to store user data, weight entries, and macro information? What would a typical document for a user's daily data look like?

It would probably start with a table of users, then link to a weight table as well as a nutrition table, weights would each only have their date/time and the weight value, while nutrition would include all of the macronutrient information, the serving, the name, as well as the date

Okay, it sounds like you're leaning towards a more relational structure even within the NoSQL context, with separate collections for users, weights, and nutrition information. This can be a good way to manage data relationships, even in MongoDB.

How would you envision linking these collections together? For example, how would you query MongoDB to retrieve all the weight entries for a specific user within a given date range, or all the nutrition logs for a particular user on a specific day? What fields would you use to establish these connections between your user, weight, and nutrition collections?

the nutrition and weight would be linked to the user individually with a user ID and then yes, we would call for nutrition of the user based on the date, and same thing for the weight

That makes sense. Using a user\_id field in both the weight and nutrition documents to link them back to the user document is a common and effective way to establish these relationships in MongoDB.

Now, let's think about querying. If you wanted to retrieve all the weight entries for a specific user within a particular date range, say between January 1st, 2023, and January 31st, 2023, how would you structure your MongoDB query? Could you give me an example of what that query might look like (in a conceptual way, you don't need to write actual code if you're not comfortable)? What operators would you use?

It has been a while since I worked with MongoDB, and I'm still in college so I'm not sure how I would structure that

That's perfectly alright! It's common to not have every detail memorized, especially if you haven't worked with a technology recently. The important thing is to understand the underlying concepts.

So, conceptually, to query for weight entries within a specific date range for a given user, you would need to:

1. **Identify the correct collection:** In this case, it would be your "weights" collection.
2. **Filter by user\_id:** You'd need to specify the user\_id of the user whose weight entries you're interested in.
3. **Filter by date:** You'd need to specify the date range. This would involve using comparison operators to find documents where the date field falls within your desired start and end dates.

While I won't ask you to write the exact MongoDB query syntax right now, does this general approach make sense? You'd essentially be telling the database to find all documents in the "weights" collection that match a specific user\_id *and* have a date value that falls within a certain range.

Have you encountered similar filtering or querying concepts in other database systems or even when working with arrays or lists in programming? Sometimes, the underlying logic is transferable even if the specific syntax differs.

It reminds me of mySQL which I have worked with recently which would be something like SELECT \* FROM weight WHERE date>=01/01/2023 and date<=01/31/2023; I don't think this is exactly right but it would look something like that

That's a great connection! You're absolutely on the right track. The fundamental idea of filtering data based on conditions is consistent across different database systems, even if the specific syntax varies.

In your MySQL example:

SQL

SELECT \* FROM weight WHERE date >= '2023-01-01' AND date <= '2023-01-31';

This query is selecting all columns (SELECT \*) from the weight table where the date column is greater than or equal to January 1st, 2023, *and* less than or equal to January 31st, 2023. You've got the core logic down perfectly!

MongoDB uses a slightly different syntax, typically involving JSON-like structures to define queries. However, the underlying principles of specifying the collection and then providing criteria (including comparison operators for the date field) remain the same.

It's excellent that you can draw parallels between different technologies. That shows a good understanding of fundamental database concepts.

Now, let's pivot slightly again. In the context of software development, what are some of the key principles or best practices you've learned that you believe are essential for writing clean, maintainable, and efficient code?

I think that some good practices are commenting, and progressive enhancement, which allow for readable code that can be expanded upon as needed.

Excellent points! Commenting your code is indeed crucial for readability and helps others (and your future self) understand the purpose and logic behind your code. Clear and concise comments can significantly reduce the time it takes to understand and maintain a codebase.

Could you elaborate on what you understand by "progressive enhancement" and why you consider it a good practice in software development, especially in the context of web applications? Perhaps you could give a brief example of how you might apply progressive enhancement in a project.

Sure, for example for progressive enhancement the idea is to start off with the MVP or minimum viable product, and then add features from there. It lets you start with just what is needed to have your program run and be useable for a variety of devices, and then add features and comforts for those devices that can support them, while ensuring the experience is still there for those who can't use the newer enhancements

That's a solid understanding of progressive enhancement! You've correctly highlighted the core idea of starting with a functional baseline that works across a wide range of environments and then layering on more advanced features and enhancements for browsers or devices that support them. This approach ensures a wider reach and a more resilient user experience.

Could you give me a specific, albeit simple, example of how you might apply progressive enhancement in a web development scenario? Perhaps think about a common UI element or interaction and how you might implement it with progressive enhancement in mind.

An example that I used recently was the details tag in html, it provides a dropdown that shows information when clicked on but it isn't revealed until it is clicked on, this isn't supported in all browsers though, and so you need to make sure that for browsers that can't use this feature they still are able to be presented the information that is held within in a readable way

That's a perfect and practical example of progressive enhancement using the <details> tag! You've correctly identified that while it provides a nice user experience in supporting browsers, it's not universally supported.

So, how would you ensure that browsers without <details> support still present the information in a readable way? What kind of fallback mechanism or approach would you implement? This is where the "enhancement" part comes in – providing a better experience where possible without breaking the fundamental functionality elsewhere.

With the details tag, a way to ensure a good experience on non-supported browsers is to make sure that the information is open by default and not hidden

That's a simple and effective fallback! By ensuring the content within the <details> tag is visible by default in browsers that don't support the tag, you maintain the core functionality of presenting the information. Users on older browsers won't get the interactive dropdown, but they won't miss out on the content either.

This demonstrates a good understanding of providing a baseline experience while leveraging newer features for enhanced functionality where available.

Now, let's switch gears one last time. Thinking about your learning journey as a new college graduate entering the software development field, what are you most excited about, and what areas do you see yourself wanting to learn and grow in the most over the next few years?

I want to grow my understanding of how code works rather than just that it works. I want to understand why something is a best practice and not just be told that it is.

That's a fantastic aspiration! Developing a deeper understanding of *why* certain practices are recommended is a hallmark of a strong and growing developer. It moves you beyond simply following instructions to making informed decisions about your code.

So, what steps do you plan to take to achieve this deeper level of understanding? Are there specific areas of software development or computer science principles that you're particularly interested in exploring further? For example, are you curious about design patterns, architectural principles, performance optimization, or something else entirely?

I think I would like to read more articles on performance optimization, as well as learn from asking those around me on the job

That's a proactive and valuable approach. Focusing on performance optimization is a great area to delve into, as it touches upon many fundamental aspects of how code runs and scales. Understanding why certain coding patterns or architectural choices impact performance can lead to significant improvements in the efficiency and responsiveness of applications.

Reading articles and engaging with experienced colleagues are both excellent ways to learn. When you're reading about performance optimization, try to not just note the "what" (e.g., "avoid unnecessary loops") but also dig into the "why" (e.g., "unnecessary loops increase the number of operations the CPU has to perform, leading to slower execution times"). Understanding the underlying principles, such as time complexity or memory management, can provide a much more robust understanding.

And you're absolutely right, leveraging the knowledge of those around you is invaluable. Don't hesitate to ask "why" when you see a particular coding style or approach being used. Most experienced developers are happy to share their reasoning and insights.

To help you get started with performance optimization, are there any specific areas that pique your interest right now? For instance, are you more curious about front-end performance (like optimizing rendering or reducing bundle sizes), back-end performance (like database query optimization or efficient algorithms), or perhaps general concepts that apply across the board? Knowing your initial interests might help point you towards some relevant resources or topics to explore.

I am mostly interested in front-end development, and I've been reading some about the CUBE CSS methodology

That's great that you're already exploring specific methodologies in front-end development! CUBE CSS (Composition, Utility, Blocks, Elements) is an interesting approach that aims to bring more predictability and maintainability to CSS.

For someone interested in front-end performance, understanding CSS methodologies like CUBE is a good starting point because well-structured CSS can contribute to better rendering performance and a smaller overall CSS footprint.

Could you tell me what aspects of CUBE CSS particularly resonate with you, and how you see it potentially contributing to better front-end performance or maintainability in larger projects? Have you had a chance to apply any of its principles in your projects so far?

I haven't used it yet, but I am working on a project where we will use it soon in the ZenGarden stylestage. I like the idea of starting at a more global level and then working down to more exact spots down the line.

That's a fantastic opportunity to get hands-on experience with CUBE CSS in the Zen Garden style stage! It's one thing to read about a methodology, and quite another to apply it to a real project. You'll likely gain a much deeper understanding of its benefits and challenges as you work with it.

Your point about starting at a more global level and then working down to more specific styles aligns well with the core principles of CUBE. This approach can indeed lead to more maintainable CSS by reducing specificity conflicts and promoting reusability of styles. It can also have performance benefits by encouraging a more streamlined and less redundant CSS codebase.

As you work on the Zen Garden project, pay attention to how the different layers of CUBE (Composition, Utility, Blocks, Elements) interact and how they help you structure your styles. Consider how this approach compares to other CSS methodologies or organizational strategies you might have used in the past. What are the advantages you notice? Are there any challenges you encounter in applying CUBE to this specific project?

This practical experience will be invaluable in solidifying your understanding of CSS architecture and its impact on front-end development. Feel free to share your insights or any questions you have as you progress with that project!

Well, we've covered a lot of ground today, from your macro tracking app to database considerations and front-end development methodologies. Do you have any questions for me about the role or the team here?

I think I'm good

Alright, it was a pleasure speaking with you today. Thank you for taking the time to share your experiences and insights. We'll be in touch regarding the next steps. Have a great rest of your day!

## Interview #3 Transcript

(Copy and paste the transcript of your AI interview here)

<https://claude.ai/share/71b5ef54-92b6-4339-a6f4-fc03894bae12>

I ran out of credits on claude.ai