URL Shortener Project - Interview Questions

1. Why did you choose Next.js for this project?

Answer:
I chose Next.js because it provided a seamless way to handle both frontend and backend logic. The API routes feature allowed me to create the URL shortening endpoint without setting up a separate server, and the file-based routing made implementing the redirection system straightforward.

Key points to mention:

- Built-in API routes for backend logic
- Server-side rendering capabilities for better SEO
- File-based routing simplified URL redirection
- Integrated frontend and backend development

2. How do you handle URL validation and security in your application?

Answer:
Currently, I validate URLs before shortening them and check for duplicate short URLs. For security, I would add:

- URL sanitization to prevent XSS attacks
- Rate limiting to prevent abuse
- Validation of the original URL's accessibility
- Implementation of a blacklist for malicious URLs

Code example:

```
// Current validation in generate/route.js
const doc = await collection.findOne({ shorturl: body.shorturl })
if (doc) {
    return Response.json({ success: false, error: true, message: 'URL already exists!' })
}
```

3. Explain your database schema and why you chose MongoDB.

Answer:
I chose MongoDB because its document-based structure perfectly suited our URL mapping needs. The schema is simple but effective, storing both the original and shortened URLs. MongoDB's performance in read operations is crucial for our redirection system.

Current schema:

```
{
    url: String,       // Original URL
    shorturl: String   // Custom short URL
}
```

Benefits of MongoDB:

- Flexible schema for easy modifications
- Good performance for read operations
- Easy to scale horizontally
- Simple to implement and maintain

4. How would you scale this application to handle millions of URLs?

Answer:
I would implement:

- Redis caching for frequently accessed URLs
- Database indexing on the shorturl field
- CDN integration for global distribution
- Horizontal scaling of the MongoDB cluster
- Load balancing for the Next.js application

Key considerations:

- Performance optimization
- Global distribution
- High availability
- Cost-effective scaling

5. What was the most challenging part of building this application?

Answer:
The most challenging part was implementing the redirection system while ensuring good performance. I had to:

- Handle edge cases for invalid URLs
- Implement efficient database queries
- Manage the MongoDB connection properly
- Ensure a smooth user experience during redirection

Code example:

```
// From [shorturl]/page.jsx
const doc = await collection.findOne({shorturl: shorturl})
if(doc){
    redirect(doc.url)
}
else{
    redirect(`${process.env.NEXT_PUBLIC_HOST}`)
}
```

Tips for Answering:

1. Always relate answers back to your actual implementation
2. Be honest about what you've implemented vs. what you would improve
3. Use specific examples from your code
4. Show your understanding of best practices
5. Demonstrate your problem-solving approach