Report on Dynamic Image Fetching for Manager's Dish Selection Without Database Load

Introduction

This report outlines an optimized approach for dynamically fetching images when a manager selects a dish name, without storing images in the database. The goal is to reduce database load while ensuring a smooth and efficient user experience.

Proposed Solution

Instead of storing images in the database, the system will utilize a **static image directory** where images are stored on the server. When a manager selects a dish, the frontend will dynamically fetch and display the corresponding image using a predefined naming convention.

Implementation Details

Backend (Spring Boot or Node.js)

- Store images in a **static folder** (e.g., static/images/dishes/).
- No image URLs are stored in the database—only dish names are recorded.
- Provide an API that dynamically constructs the image URL based on the dish name.
- If an image does not exist, a **default image** will be served instead.

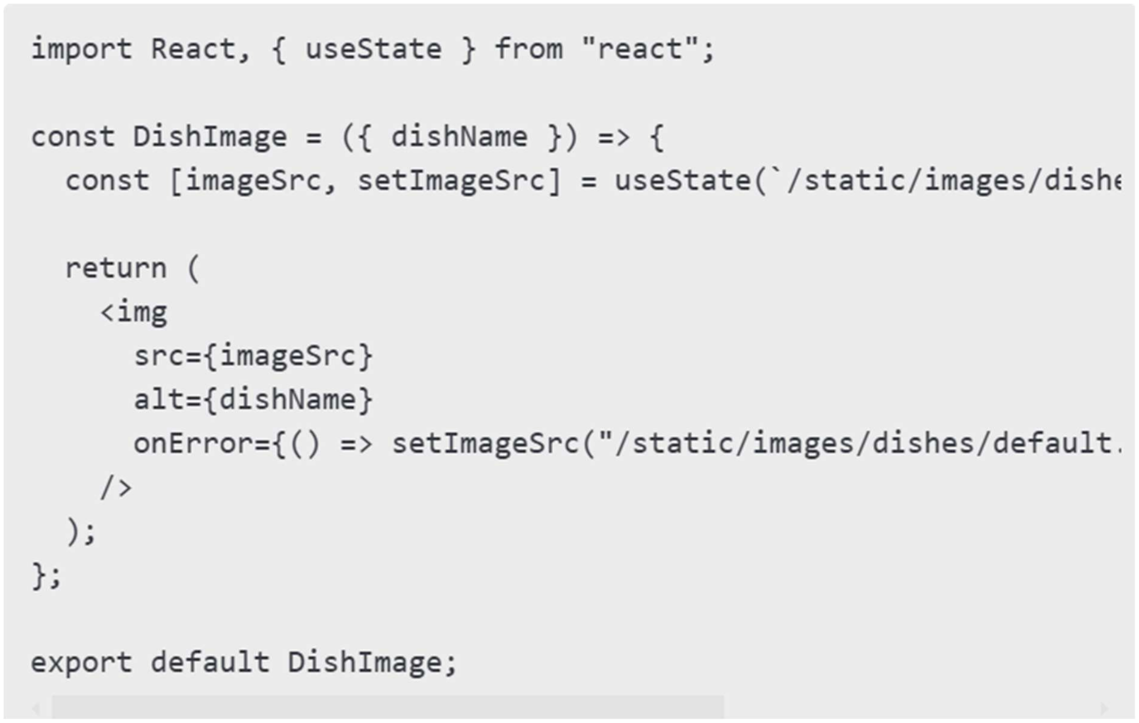
Sample API Response

```json
{
  "dishName": "Pizza",
  "imageUrl": "https://yourserver.com/images/dishes/pizza.jpg"
}
```

Frontend (ReactJS)

- When a **manager clicks on a dish name**, fetch the dynamically generated image URL.
- Use an **onError fallback mechanism** to replace missing images with a default image.
- Ensure images are loaded directly from the static directory instead of querying the database.

Sample ReactJS Implementation



```
import React, { useState } from "react";

const DishImage = ({ dishName }) => {
  const [imageSrc, setImageSrc] = useState(`/static/images/dishe

  return (
    <img
      src={imageSrc}
      alt={dishName}
      onError={() => setImageSrc("/static/images/dishes/default.
    />
  );
};

export default DishImage;
```

Advantages of This Approach

- **No Database Storage for Images** – Only dish names are stored, reducing database load.
- **Faster Performance** – Static images are served directly without additional queries.
- **Scalability** – Can handle a large number of dish images without affecting backend performance.
- **Simplified Maintenance** – New images can be added or replaced by updating the static folder.
- **Enhanced User Experience** – Instant image loading when a dish is selected.

Conclusion

This method efficiently handles image fetching when a manager selects a dish name, without relying on database storage for images. By leveraging a static image directory and dynamic URL construction, we ensure minimal backend load while maintaining high performance and scalability.