Artificial Intelligence (CSE5005)

Mini Project Presentation (13th March 2025)

Creating a Customer Recommendation System based on an Ecommerce platform

By

Name	Roll Number
Vishnu Priya	20242AIE0007
Yerrabathina Muni Sekhar	20242AIE0008
Yenimireddy Venkata Ramireddy	20242AIE0009
Anas Muhammed Sahil	20242AIE0010
Rojin K Varughese	20242AIE0011

Under the supervision of

Dr. Raja Jitendra Nayaka

Assisstant Professor

Department of Computer Science & Engineering

,

PRESIDENCY UNIVERSITY
GAIN MORE KNOWLEDGE REACH GREATER HEIGHTS
Private University Estd. in Karnataka State by Act No. 41 of 2013

PRESIDENCY GROUP OVER 40 YEARS OF ACADEMIC WISDOM

Problem Statement

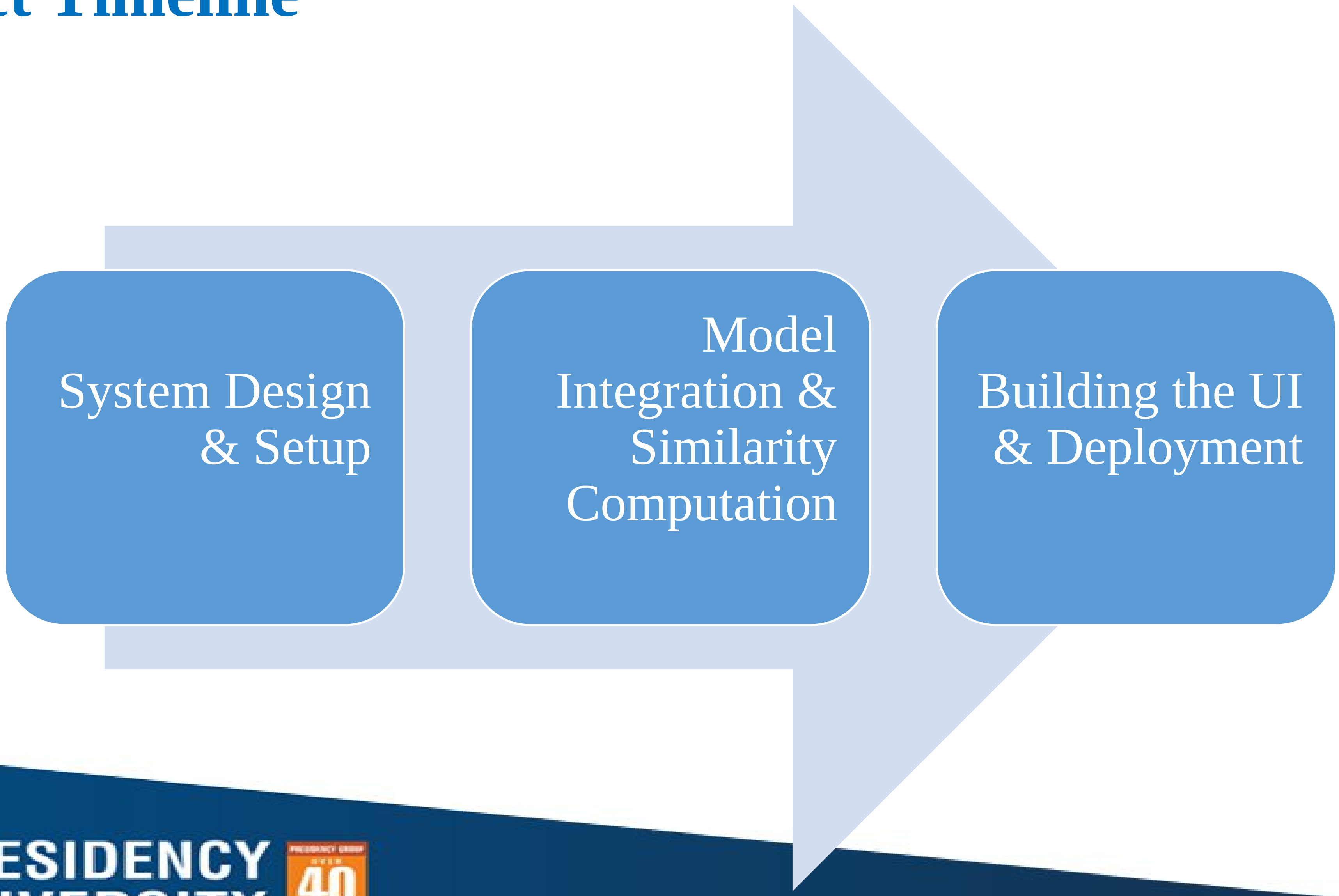
- Online shoppers often struggle to find relevant products.
- Traditional recommendation systems rely on explicit user data, which may be unavailable.
- Need a **lightweight, client-side recommendation system** that runs in the browser.
- Goal: **Use image-based similarity to recommend visually similar products**.

Comprehensive plan to address the Problem Statement

Approach:

- **Use FakeStoreAPI** as a mock e-commerce database.
- **MobileNet (pre-trained CNN model)** to extract product image features.
- **TensorFlow.js** to compute similarity between product images in the browser.
- **Deploy as a lightweight web application with real-time recommendations.**

Project Timeline



System Design & Setup

Model Integration & Similarity Computation

Building the UI & Deployment

Phase 1 - System Design & Setup

- Define requirements & architecture.
- Set up TensorFlow.js
- Set up FakeStoreAPI.
- Load & preprocess product images.

Phase 2 - Model Integration and similarity computation

- Use MobileNet to extract image embeddings.
- Compute cosine similarity between products.
- Generate list of similar products

Phase 3 - Building the UI & Deployment

- Create a simple UI with HTML, BulmaCSS
- Deploy as a browser based app.

Implementation

The project is implemented in JavaScript with an online API to generate products and a “MobileNET” is used to create a list of similar products.

Implementation - Tensorflow.js

- A **JavaScript library for machine learning**, enabling AI models to run directly in the browser.
- Supports **pre-trained models** like MobileNet and allows **on-the-fly model training**.
- Uses **WebGL acceleration** for efficient performance without requiring a backend server.

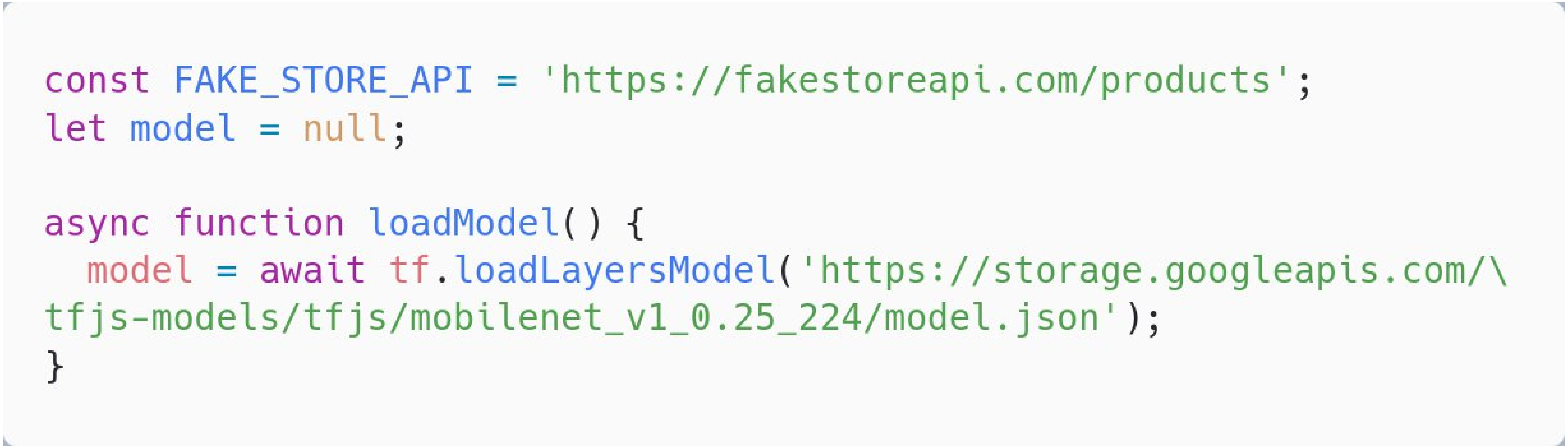
Implementation - MobileNET

- A **lightweight, pre-trained CNN model** optimized for mobile and web applications.
- Uses **depthwise separable convolutions** to reduce computational cost while maintaining accuracy.
- Ideal for **feature extraction** in image-based machine learning tasks like similarity detection

Implementation - FakestoreAPI

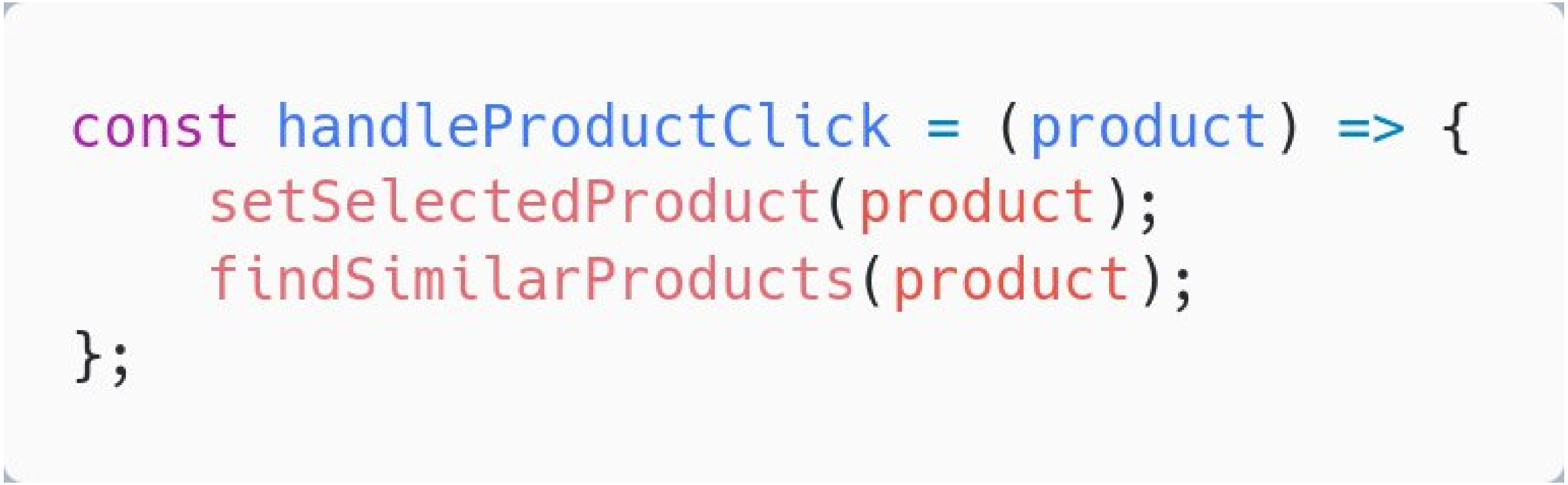
- A **free, fake e-commerce REST API** providing product data like images, titles, prices, and categories.
- Useful for **prototyping and testing recommendation systems** without needing real product databases.
- Supports **simple API calls (in JSON)**, which makes it easy to integrate into JavaScript-based applications.

Implementation - Initial setup



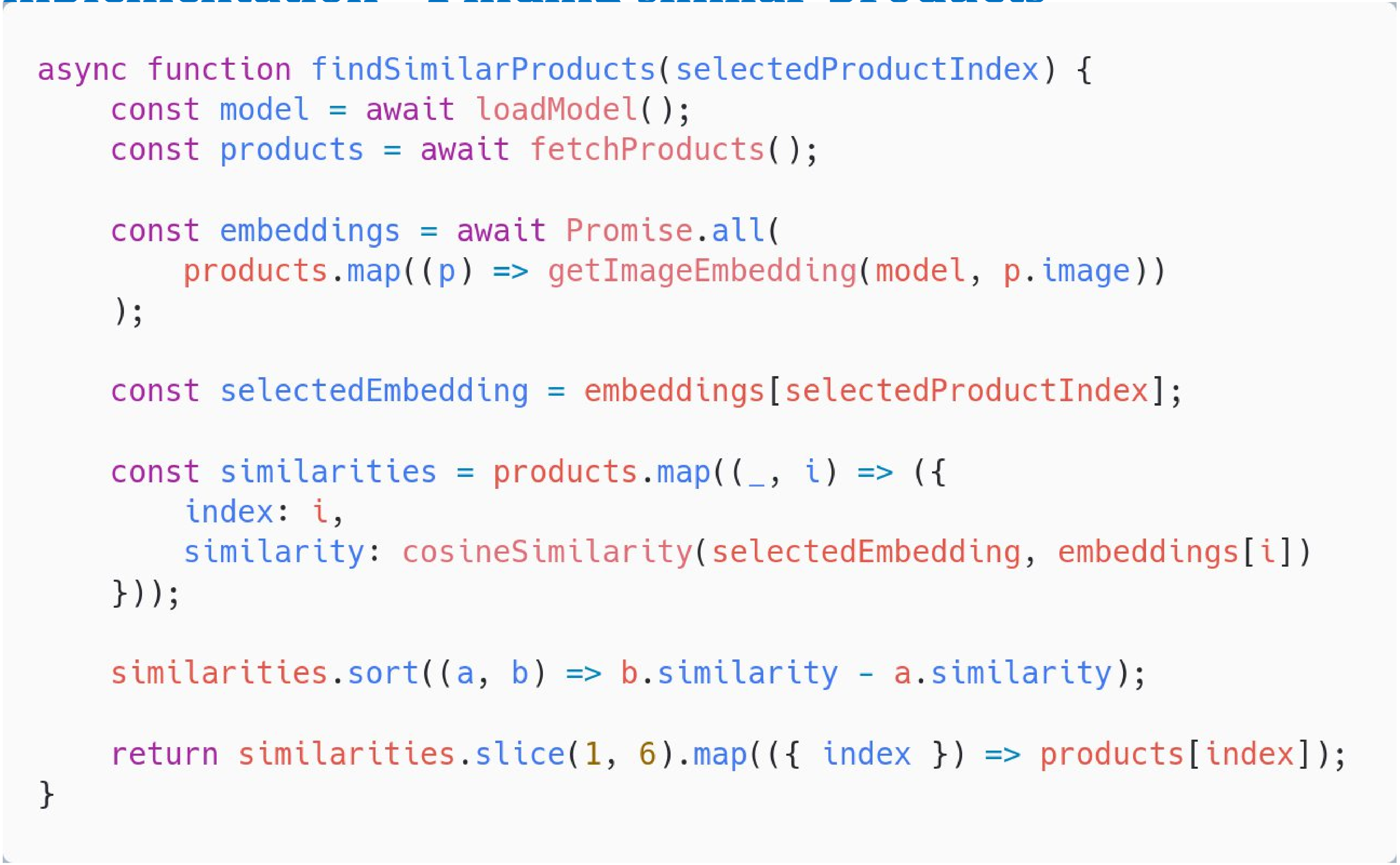
```
const FAKE_STORE_API = 'https://fakestoreapi.com/products';
let model = null;

async function loadModel() {
  model = await tf.loadLayersModel('https://storage.googleapis.com/\
tfjs-models/tfjs/mobilenet_v1_0.25_224/model.json');
}
```

Implementation - Finding similar products



```
const handleProductClick = (product) => {
    setSelectedProduct(product);
    findSimilarProducts(product);
};
```

Implementation - Finding similar products



```
async function findSimilarProducts(selectedProductIndex) {
    const model = await loadModel();
    const products = await fetchProducts();

    const embeddings = await Promise.all(
        products.map((p) => getImageEmbedding(model, p.image))
    );

    const selectedEmbedding = embeddings[selectedProductIndex];

    const similarities = products.map((_, i) => ({
        index: i,
        similarity: cosineSimilarity(selectedEmbedding, embeddings[i])
    }));

    similarities.sort((a, b) => b.similarity - a.similarity);

    return similarities.slice(1, 6).map(({ index }) => products[index]);
}
```

Expected Challenges & Mitigation

Problem	Solution
Performance issues in browser	Optimize with WebGL-backed TF.js execution
Low-quality product images	Preprocess images before extracting features
Scalability concerns	Use approximate nearest neighbors for large datasets

Conclusion & Next Steps

Built a lightweight, AI-powered recommendation system using Tensorflow.js and MobileNet.

Next Steps:

- Implement in a demo e-commerce store.
- Gather feedback and optimize performance.
- Extend with additional ML models.

Q&A

Thank you !!