CODE ARCHITECT

Team Members

Gokul P V

Anagha M B

Chandu Dadi

Meet Code Architect!

Picture this:

You're about to embark on an epic journey. You've meticulously planned your route, studied the terrain, and prepared for challenges. But no matter how well you've plotted your course, the success of your adventure depends on what you pack in your backpack. Too much weight can slow you down, while the wrong equipment might leave you unprepared for obstacles ahead.

Our Code Architect is like having an experienced guide who knows exactly what you'll need for your journey.

Based on your problem statement and proposed architecture - your 'journey map' - our tool recommends the best 'gear' (applications, platforms, and tech stack) to pack. This way, you're prepared for any challenge and can reach your destination efficiently and effectively. Just as you wouldn't set out on a hike without the right gear, don't embark on a tech project without the optimal tech stack.

Let our Code Architect, guide you to success.

Unveiling the Power of Efficient Designs

What happens when we don't!

- **Facebook's** 2018 Data Breach: In 2018, Facebook experienced a data breach that affected around 50 million users due to a flaw in their system architecture. The company's "View As" feature was exploited, allowing hackers to steal Facebook access tokens. This breach resulted in a significant loss in user trust and a 2.6% drop in stock value, translating to a loss of around $13 billion in market capitalization.

- **Capital One's** 2019 Data Breach: Capital One, a well-known bank that heavily uses data science and IT for its operations, suffered a massive data breach in 2019. The breach was due to a misconfigured firewall in their cloud-based system architecture. It resulted in a loss of personal data for over 100 million customers and cost the company an estimated $150 million, aside from the significant damage to their reputation.

- **Equifax's 2017** Data Breach: Credit reporting company Equifax suffered a massive data breach in 2017 due to a vulnerability in their system architecture. The breach led to the exposure of personal information of 147 million people, including their social security numbers. The fallout from the breach cost Equifax over $1.4 billion in cleanup costs and a significant loss in market value.

Journey from Current scenario to the Desired state

Background

- The technology landscape is continually evolving with new frameworks, tools, and languages, making the task of selecting an appropriate tech stack complex.
- Tech stack choice greatly impacts the software product's performance, scalability, and maintainability, making it a crucial decision.
- Effective selection requires deep understanding of project requirements and foresight to predict long-term performance of chosen technologies.

Current Challenges

- The complexity of tech stack selection often leads to uncertainty and risk of system incompatibility.
- Analysing and comparing different tech stacks is time-consuming and resource-intensive.
- Inappropriate tech stack selection can lead to scalability issues, maintenance challenges, and project setbacks.
- Keeping up-to-date with rapidly evolving technology trends and having the necessary technical expertise for selection is a recurring challenge.

Desired State

- We introduce Code Architect, a Gen-AI-based tool designed to analyse a project's current architecture and recommend an optimized tech stack.
- Code Architect intelligently assesses existing architecture, identifies potential improvements, and suggests efficient tech stack changes, considering Org level constraints as well.
- It ensures improved system compatibility, scalability, and maintainability.
- Code Architect stays updated with the latest technology trends, ensuring your project is always equipped with the most suitable and current tech stack.

Solution Details

CV BASED

ENTITY DETECTION

1

USER INPUT

Technical Architecture in the form of Diagram as code.

Customizable Parameters –

- Data volume
- Data type
- Processing type – Real time/Batch
- User Volume
- Optimization parameters – Cost, Time, Efficiency
- Any additional Comments

FINE TUNING

Fine tuning of LLAMA 3.2 1B Instruct model is performed with 61 handcrafted training samples. Supervised finetuning was performed using HuggingFace and Unsloth.

2

LLM BASED MODEL

The user inputs are passed into the LLM model. LLAMA 3.2 1B Instruct model is being used in the backend.

Based on our prompting the finetuned LLM will return the desired output.

3

OUTPUT

The output from the LLM model is passed through a json parser to provide the desired result to the UI

Output will include –

- Recommendations and its reasons
- These recommendations can be utilized to generate the new architecture.

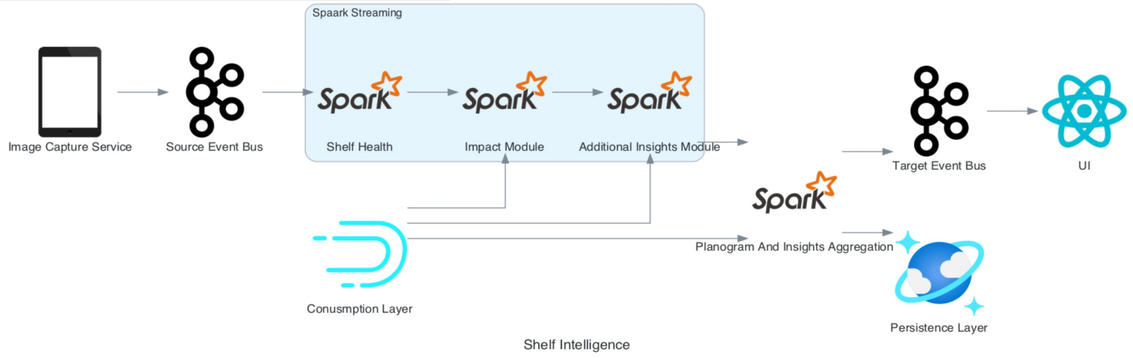
Future Scope

Demo Video Here

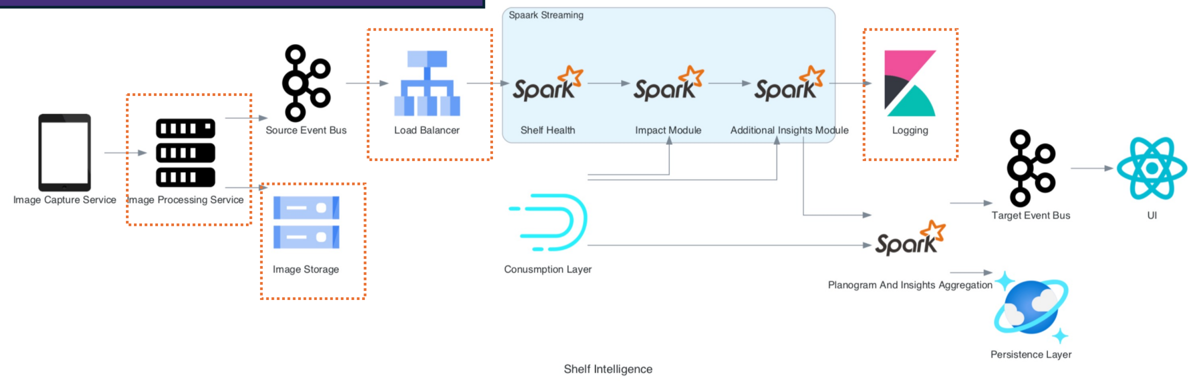
https://huggingface.co/spaces/gokul-pv/CodeArchitect

Sample results

Input Diagram



Output Diagram



Recommendations

The current architecture seems to be well thought out. However, there are a few areas that could be improved.

Image Processing: The images captured by the cameras are likely to be large and processing them in real-time could be challenging. Consider using a dedicated image processing service or library to reduce the size and complexity of the images before they are queued in Kafka. This could improve the speed and efficiency of the system.

Data Storage: The Druid DB is used to fetch data for calculations, but it might not be the best choice for storing image data. Consider using a more suitable database for storing and retrieving large amounts of image data, such as Google Cloud Storage.

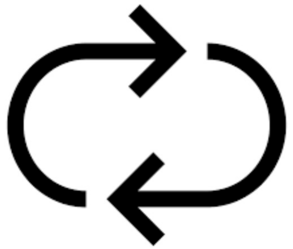
Fault Tolerance: The architecture should be designed to handle failures gracefully. Consider adding a mechanism to handle failures in the Kafka queues or the Spark Streaming modules. This could be achieved by replicating the state of the system and recovering from a replicated state in case of a failure.

Scalability: The system should be able to handle the load of thousands of stores and cameras. Consider using a load balancer to distribute the load across multiple instances of the Spark Streaming modules. This could improve the speed and accuracy of the system.

Cost Optimization: The use of cloud services could lead to high costs. Consider using spot instances or reserved instances to reduce costs. Additionally, consider using auto-scaling to dynamically adjust the number of instances based on the load. This could significantly reduce costs during off-peak hours.

Debugging: The system should be easy to debug. Consider adding logging and monitoring services to the architecture. This could help in identifying and resolving issues quickly.

Future Roadmap



Feedback Loop Integration

Develop a robust feedback system that users can use to rate and review the recommended tech stack. This data can be used to continuously improve the tool's recommendations.

CV based entity detection

Usage of computer vision-based extraction of architecture diagrams for better user experience and ease of use

Cost Estimation feature

Adding a cost estimation feature that can provide an approximate budget based on the recommended tech stack will

Thank You!