

Efficient Road Repairs System

with YOLOv8, XILINX, phi-3.5, FAISS and EPCIS

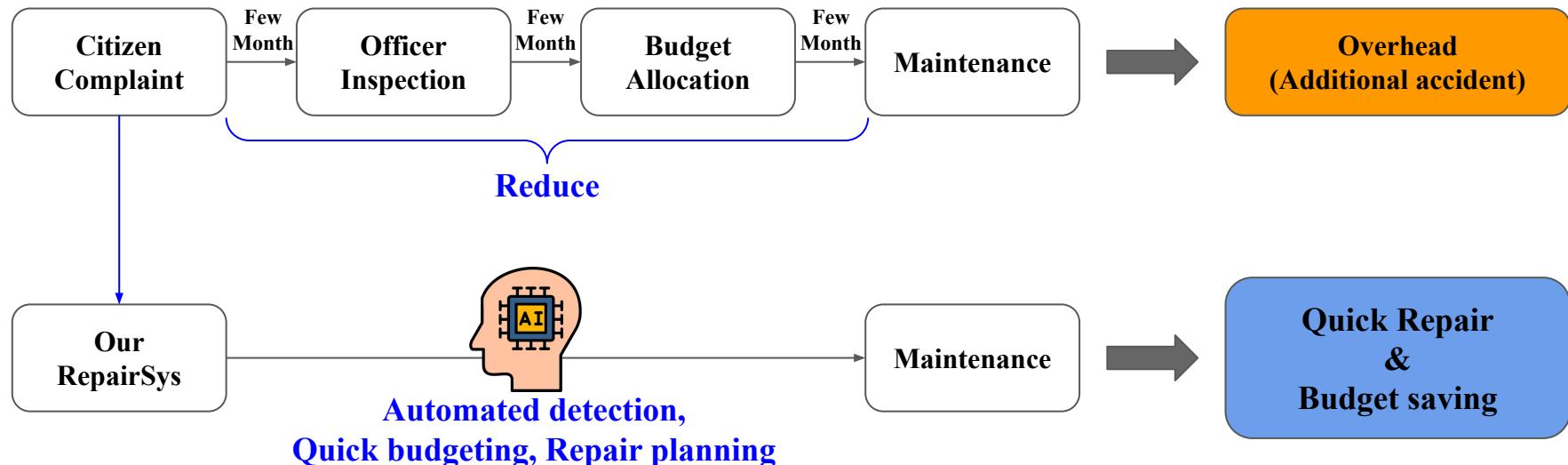
Team #1, Final project presentation

Donghee Han, Jungjae Lee, Seungjae Lim, Seungmin Yang, Zhaoyan Wang

<https://github.com/SeungjaeLim/Efficient-Road-Repairs-System>

Motivation Recap

[AS-IS] Lack of automation system (human-based)



[TOBE] AI-based automated Road Repairs System



Train Road Damage Detection
YOLOv8 with RDD2022



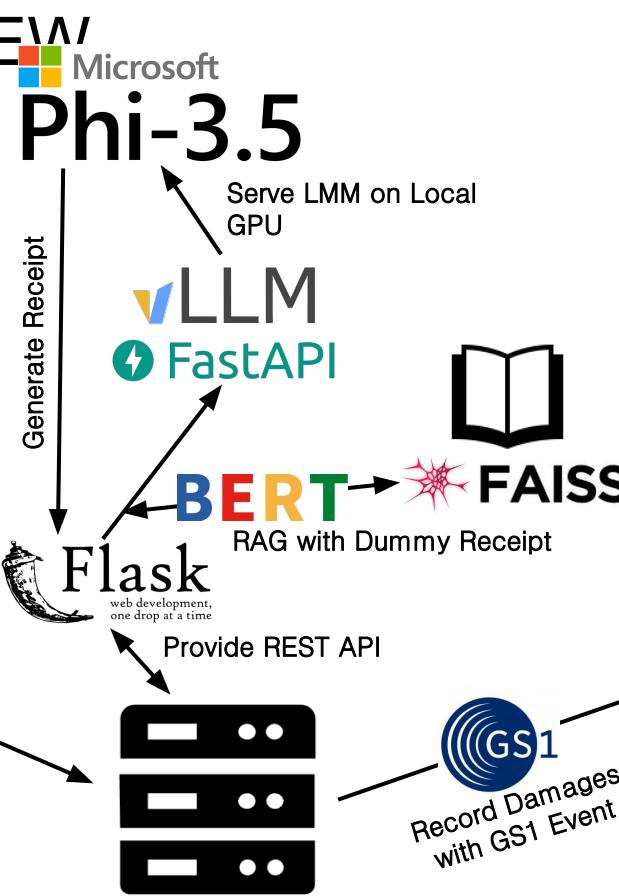
Porting YOLOv8
to XILINX FPGA



Real-time road
damage detection on
car



Communicate
with Socket



Provide to Gov.
with Web-app





Train Road Damage Detection
YOLOv8 with RDD2022



Porting YOLOv8
to XILINX FPGA



Real-time road
damage detection on
car



Communicate with Socket

Real-time Road Damage Detection on FPGA



Phi-3.5

Serve LMM on
Local GPU

LLM
FastAPI

BERT

RAG with Dummy Receipt



Provide REST API



FAISS



Record Damages
with GS1 Event



EPCIS

Store GS1 Data
MongoDB
redis



한국도로공사



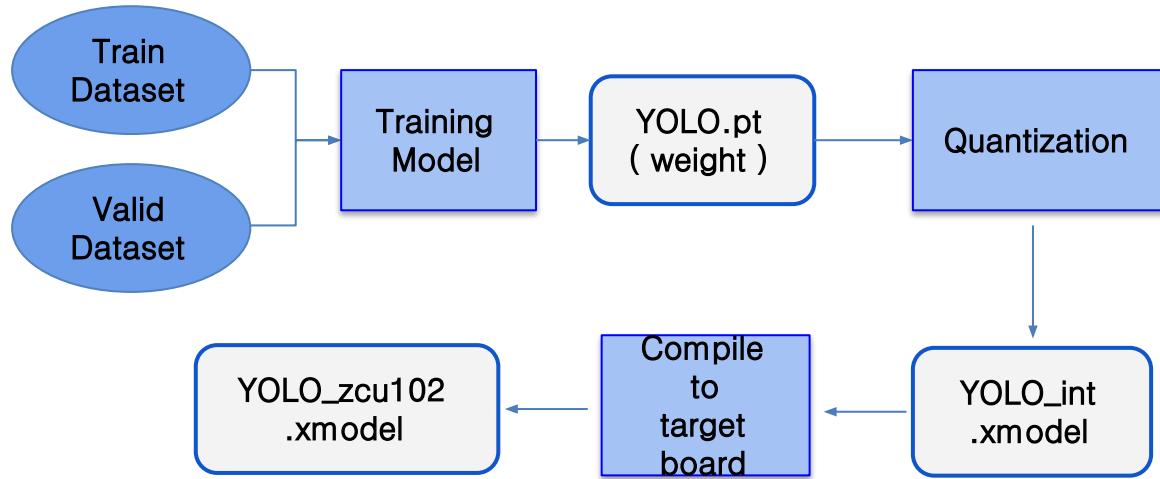
Provide to Gov.
with Web-app



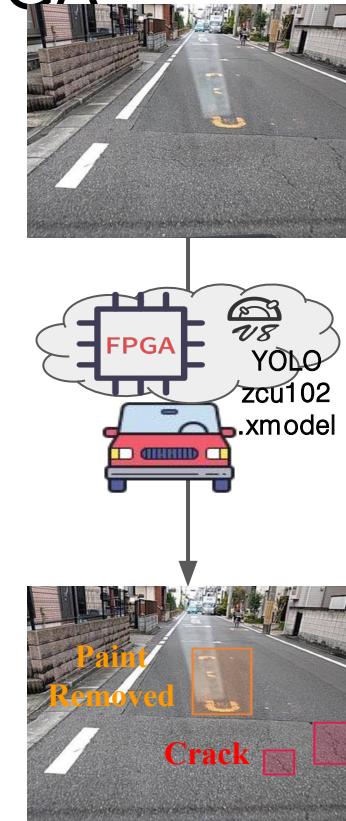
GS1 Load Damages
with GS1 Query

2. Real-time Road Damage Detection on FPGA

Detection model build-up pipeline



Dataset: Road Damage Detection Challenge 2022 dataset (RDD 2022)
Detection model: YOLOv8
FPGA: Xilinx FPGA ZCU 102



2.1 YOLOv8-Enabled Road Damage Detection

Dataset: Road Damage Detection Challenge 2022 dataset (RDD 2022)

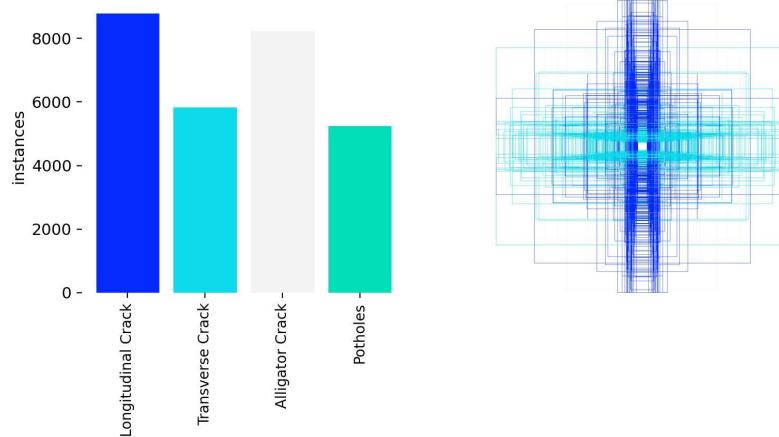
Model trained: YOLOv8m



Samples utilized

Asian country division:

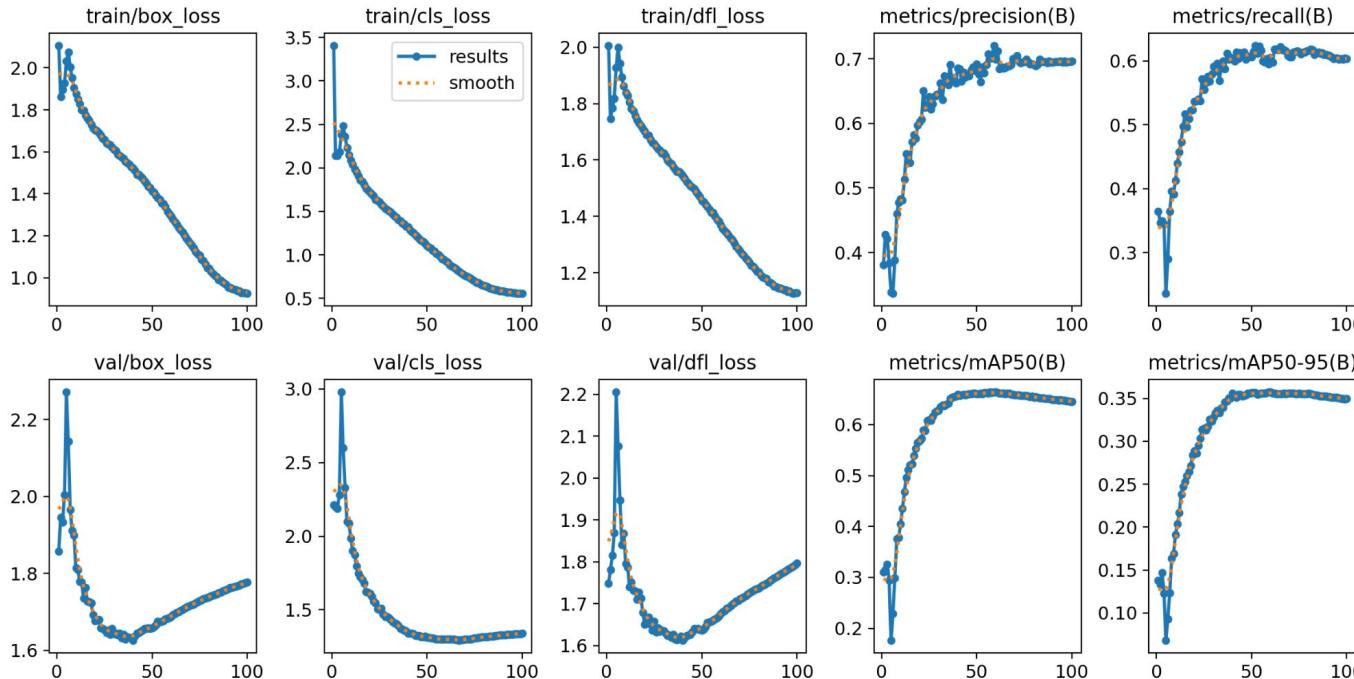
China Drone & China MotorBike & India & Japan



- Dataset conversion from PascalVOC to YOLOv8 format.
- Train and val dataset split: 9:1
- Training & Evaluating YOLOv8 ultralytics
- Image cropping on detected objects

2.1 YOLOv8-Enabled Road Damage Detection

YOLOv8m Model Performance



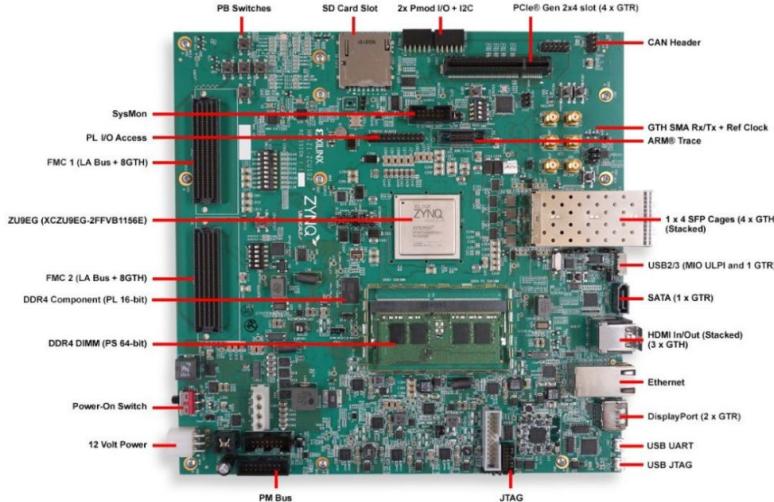
2.1 YOLOv8-Enabled Road Damage Detection

Val-batch snapshot:



2.2 Yolo Porting

Xilinx FPGA ZCU 102



Chip	Xilinx Zynq UltraScale+ MPSoC (XCZU9EG-2FFVB1156E)
Processor	Quad Core Arm Cortex-A53(1.5Ghz) / Dual Core Cortex-R5(600Mhz)
GPU	Dual core Mali MP2 GPU(up to 667Mhz)
Memory	4GB DDR4
System Logic Cells	600,000

2.2 Yolo Porting

Quantization

- Reduce Memory Usage, Computation on FPGAs
- Performing INT 8-bit quantization

Compilation

- Compile to the architecture of the FPGA (ZCU102)
- Optimized for DPU utilization
- Change activation function for DPU compatibility
(Silu → Hardswish)
- YOLO_post_process

Implement operations not supported by the DPU and post-process the resulting values into an interpretable form.

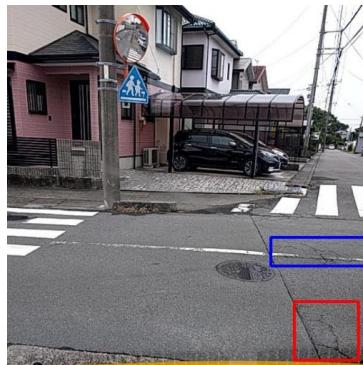
Run Model

- Run the model on the target board (ZCU102)
- Single Thread
 - Model load : 3.707s
 - Data preprocess : 0.103s
 - Model run : 0.117s
 - Post process : 0.224s

```
void yolov8_post_process(const std::vector<vart::TensorBuffer *> &output_tensor_buffers)
{
    // std::vector<vitis::ai::library::OutputTensor> detect_output_tensors;
    std::vector<vart::TensorBuffer *> detect_output_tensors;

    int num_classes = 6;

    for (auto i = 0u; i < detect_layer_name.size(); i++)
    {
        for (auto j = 0u; j < output_tensor_buffers.size(); j++)
        {
            auto output_tensors_unsorted = output_tensor_buffers[j]->get_tensor();
            auto pos = output_tensors_unsorted->get_name().find(detect_layer_name[i]);
            if (pos != std::string::npos && pos + detect_layer_name[i].size() == output_ten
```



	YOLOv8-m
Single Thread	2.5 FPS
Multi Thread	10-12 FPS

Bottleneck : dash cam = 30FPS

- A crack is detected for at least 3 consecutive frames.

- Processing only once every 3 frames allows for real-time processing without bottlenecks.

2.3. Socket Real-time Detection

- We used socket to communicate with the server to send the detected results to the server in real time.
- Original Image + text (label + bounding box coordinates)

Client (FPGA)

Server

```
std::vector<unsigned char> buffer;
cv::imencode(".jpg", image, buffer);
std::string imageData = std::string(buffer.begin(), buffer.end());

int imageSize = imageData.size();
int textSize = textData.size();

send(sock, &imageSize, sizeof(imageSize), 0);
send(sock, &textSize, sizeof(textSize), 0);
send(sock, imageData.c_str(), imageSize, 0);
send(sock, textData.c_str(), textSize, 0);

close(sock);

return 0;
}

std::string textData = "";
for (int i = 0; i < res_vec.size(); i++) {
    int label = (int)(res_vec[i].label);
    float score = (float)(res_vec[i].score);

    textData += std::to_string(label) + " ";
    for (int j = 0; j < 4 ; j++ ) { textData += std::to_string((int)res_vec[i].box[j]) + " " ; }
    textData += "\n";
}
```

Image Data

```
try:
    image_size_data = conn.recv(4)
    image_size = int.from_bytes(image_size_data, byteorder='little')

    text_size_data = conn.recv(4)
    text_size = int.from_bytes(text_size_data, byteorder='little')

    image_data = receive_all(conn, image_size)
    if image_data is None: print("Failed to receive the complete image data.")

    text_data = receive_all(conn, text_size)
    if text_data is None: print("Failed to receive the complete text data.")

    decoded_text = text_data.decode('utf-8')
    print("Received text:", decoded_text)

    save_image(image_data)
```

Text Data

2 REST API Server



Train Road Damage Detection
YOLOv8 with RDD2022



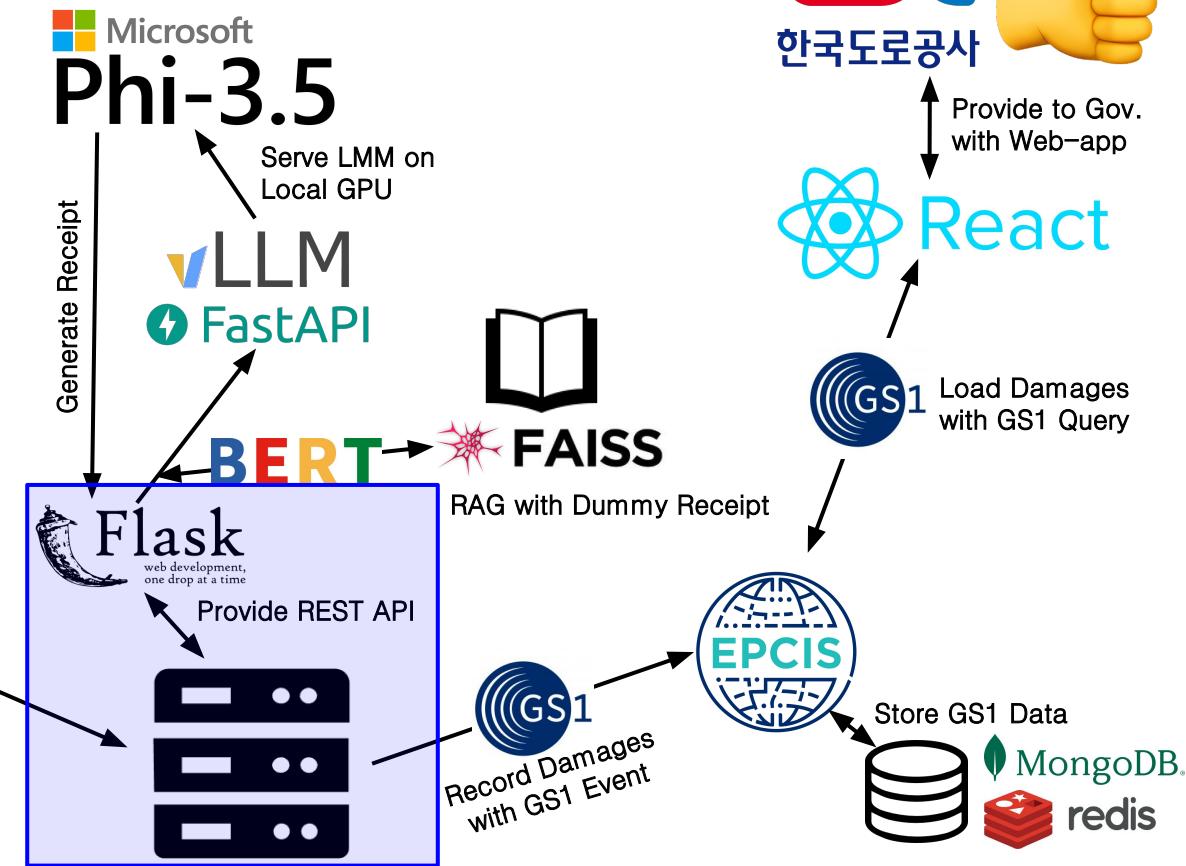
Porting YOLOv8
to XILINX FPGA



Real-time road
damage detection on
car

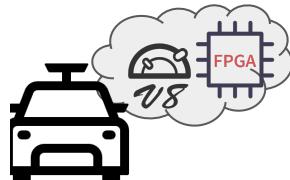
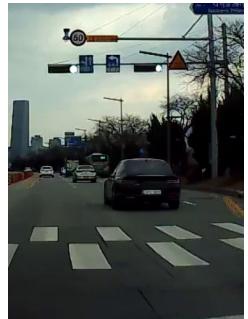


Communicate
with Socket

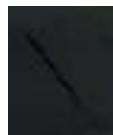


3. REST API Server

Microsoft
Phi-3.5



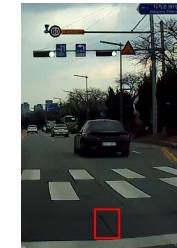
Send to LLMEngine with RAG &
Img
FAISS
BERT



Post-processing
Generated Receipt



GET /static/images
To Serve Boxed Img



POST /process
To get img & json



web development,
one drop at a time

Request to GS1 Format



3. REST API Server

Label and Crack frame

```
{  
    "label": 0,  
    "filename": "frame_0916.png",  
    "x1": 759.98,  
    "y1": 655.26,  
    "x2": 799.25,  
    "y2": 702.24  
},
```

Receipt from RAG system

```
{  
    "Damage Type": "longitudinal crack",  
    "Width": 39.27,  
    "Height": 46.98,  
    "Repair Items": [  
        {  
            "ItemDescription": "Surface Cleaning and Preparation",  
            "Quantity": 1,  
            "UnitPrice": 144.66,  
            "TotalPrice": 144.66  
        },  
    ],  
}
```

LMM-based Generated Receipt

```
{  
    "ItemDescription": "Traffic Management (Signs/Barriers)",  
    "Quantity": 1,  
    "UnitPrice": 101,  
    "TotalPrice": 101  
},  
{  
    "Repair Cost": 1880.62,  
    "Dimensions": {  
        "Width": 39.26999999999998,  
        "Height": 46.98000000000002  
    }  
}
```

3. REST API Server

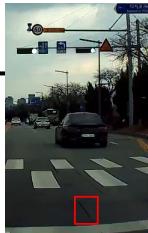
BERT
Download Pretrained
bert-base-nli-mean-toke
ns

FAISS
Index dummy receipts



Run Server

GET /static/images
To Serve Boxed Img



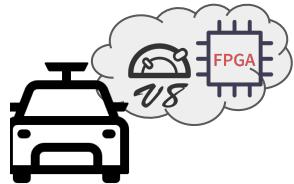
POST /process
To get img & json



```
{  
    "label": 0,  
    "filename": "frame_0916.png",  
    "x1": 759.98,  
    "y1": 655.26,  
    "x2": 799.25,  
    "y2": 702.24  
},
```

```
Loading faiss.  
Successfully loaded faiss.  
PyTorch version 2.5.1 available.  
Use pytorch device name: cuda  
Load pretrained SentenceTransformer: bert-base-nli-mean-tokens  
Batches: 100% [██████████] 4/4 [00:00<00:00, 6.29it/s]  
Dummy data and FAISS index loaded successfully.  
-----  
Flask Server is listening on port 5000  
-----  
Model ID: microsoft/Phi-3.5-vision-instruct  
-----  
* Serving Flask app 'config'  
* Debug mode: off  
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.  
* Running on all addresses (0.0.0.0)  
* Running on http://127.0.0.1:5000  
* Running on http://172.17.0.2:5000  
Press CTRL+C to quit  
[Request] Serving file from: /workspace/flask-server/CROPPED_IMAGES/boxed_20241213_133038.png  
172.17.0.1 - - [13/Dec/2024 13:31:10] "GET /static/images/boxed_20241213_133038.png HTTP/1.1"  
200 -  
[Request] Serving file from: /workspace/flask-server/CROPPED_IMAGES/boxed_20241213_133026.png  
172.17.0.1 - - [13/Dec/2024 13:31:10] "GET /static/images/boxed_20241213_133026.png HTTP/1.1"  
200 -  
[Request] Serving file from: /workspace/flask-server/CROPPED_IMAGES/boxed_20241213_133032.png  
[Request] Serving file from: /workspace/flask-server/CROPPED_IMAGES/boxed_20241213_123346.png  
172.17.0.1 - - [13/Dec/2024 13:31:10] "GET /static/images/boxed_20241213_133032.png HTTP/1.1"  
200 -  
[Request] Serving file from: /workspace/flask-server/CROPPED_IMAGES/boxed_20241213_123341.png  
172.17.0.1 - - [13/Dec/2024 13:31:10] "GET /static/images/boxed_20241213_123341.png HTTP/1.1"  
304 -  
[Request] Serving file from: /workspace/flask-server/CROPPED_IMAGES/boxed_20241213_123335.png  
172.17.0.1 - - [13/Dec/2024 13:31:10] "GET /static/images/boxed_20241213_123335.png HTTP/1.1"  
304 -  
[Request] Request received  
Batches: 100% [██████████] 1/1 [00:00<00:00, 54.35it/s]  
[Retrieval] Retrieved indices: [92, 62, 44]  
HTTP Request: POST http://localhost:8082/v1/chat/completions "HTTP/1.1 200 OK"  
172.17.0.1 - - [13/Dec/2024 13:31:23] "POST /process HTTP/1.1" 200 -  
[Request] Request received  
Batches: 100% [██████████] 1/1 [00:00<00:00, 93.68it/s]  
[Retrieval] Retrieved indices: [32, 37, 13]  
HTTP Request: POST http://localhost:8082/v1/chat/completions "HTTP/1.1 200 OK"  
172.17.0.1 - - [13/Dec/2024 13:31:28] "POST /process HTTP/1.1" 200 -  
[Request] Request received  
Batches: 100% [██████████] 1/1 [00:00<00:00, 62.10it/s]  
[Retrieval] Retrieved indices: [78, 15, 13]  
HTTP Request: POST http://localhost:8082/v1/chat/completions "HTTP/1.1 200 OK"
```

3. REST API Server



```
@app.route('/process', methods=['POST'])
def process_image_and_json():
    """
Handle the image processing and data enrichment logic:
1. Validate and parse the input (image and JSON).
2. Process the image (crop, resize, box drawing).
3. Interact with LMM (OpenAI) service to get repair suggestions.
4. Calculate costs and finalize the output JSON.
5. Send event data to EPCIS server.
6. Return the final JSON response.

:return: A JSON response with final results.
    """
```



```
@app.route('/static/images/<path:filename>', methods=['GET'])
def serve_image(filename):
    """
Serve images from the CROPPED_IMAGES directory.

:param filename: The name of the image file to serve.
:return: The requested file or a 404 if not found.
    """
```



```
def send_to_epcis(parsed_result, lmm_input, boxed_image_path):
    """
Send event data to the EPCIS server.

:param parsed_result: The final dictionary containing cost info and damage details.
:param lmm_input: The dictionary with dimension and damage type.
:param boxed_image_path: Path to the boxed image file.
:return: The text response from the EPCIS server.
:raises Exception: If the EPCIS server returns a non-success status code.
    """
```

RAG Pipeline



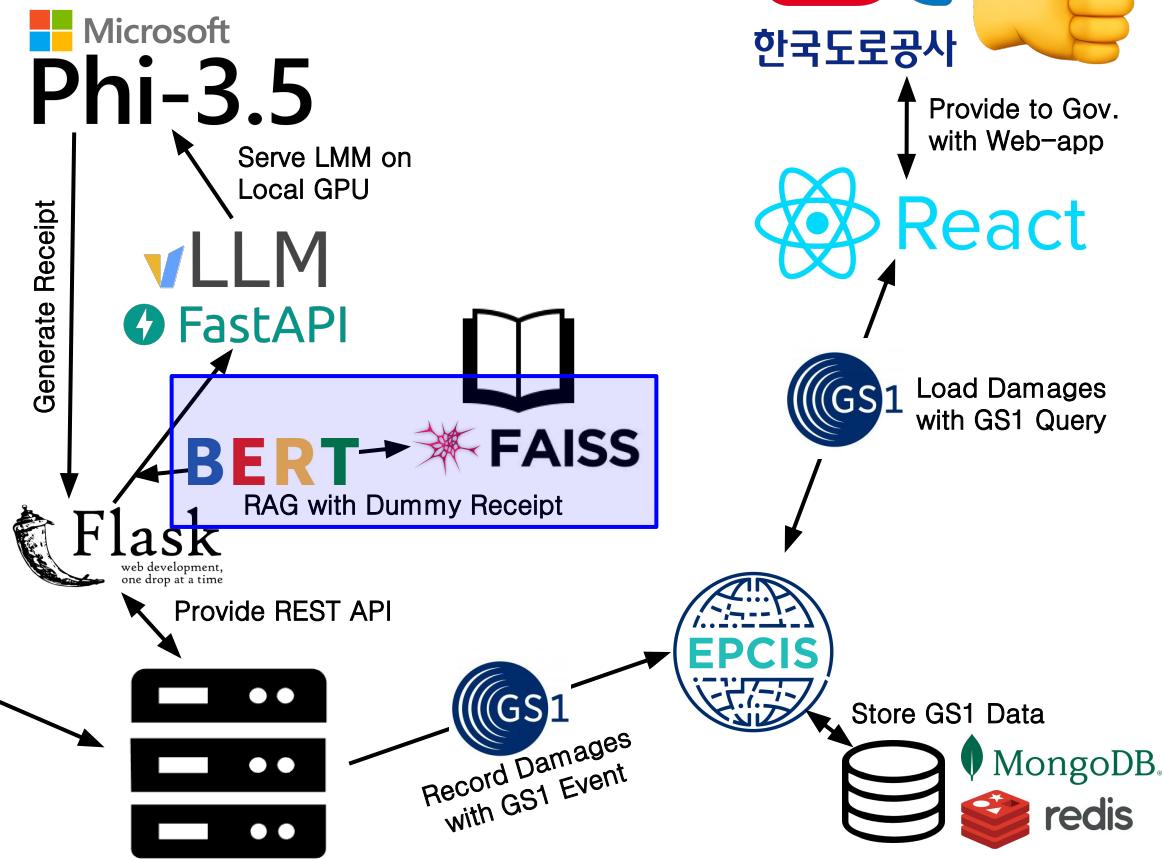
Train Road Damage Detection
YOLOv8 with RDD2022



Porting YOLOv8
to XILINX FPGA



Real-time road
damage detection on
car



ROAD REPAIR Pipeline

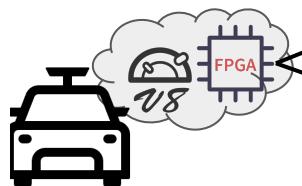
BERT

```

{
  "Damage Type": "longitudinal crack",
  "Width": 39.27,
  "Height": 46.98,
  "Repair Items": [
    {
      "ItemDescription": "Surface Cleaning and Preparation",
      "Quantity": 1,
      "UnitPrice": 144.66,
      "TotalPrice": 144.66
    },
    ...
    {
      "ItemDescription": "Traffic Management (Signs/Barriers)",
      "Quantity": 1,
      "UnitPrice": 101,
      "TotalPrice": 101
    }
  ],
  "Repair Cost": 1880.62,
  "Dimensions": {
    "Width": 39.26999999999999,
    "Height": 46.98000000000002
  }
}

```

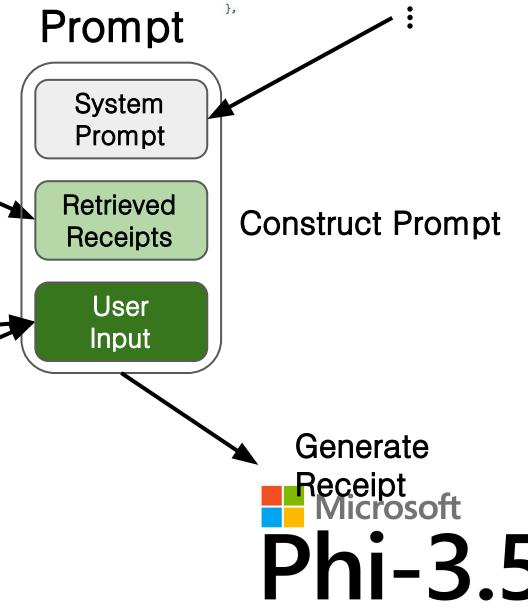
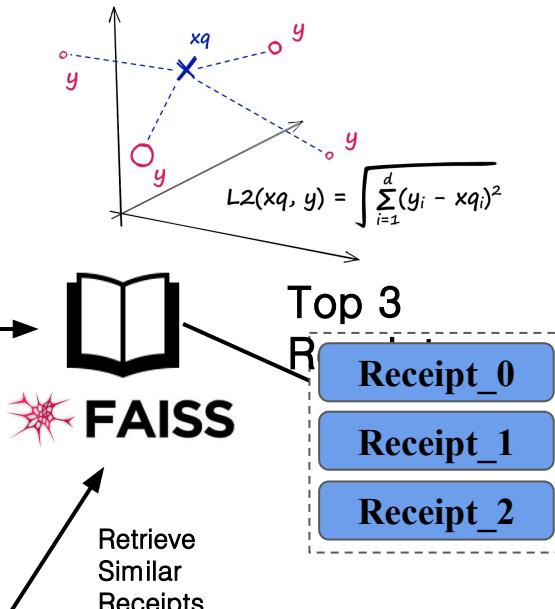
x 10000
Receipt Database



```
{
  "label": 0,
  "filename": "frame_0916.png",
  "x1": 759.98,
  "y1": 655.26,
  "x2": 799.25,
  "y2": 702.24
},
```



Image file



You are a road crack repair cost estimator agent. Your task is

Input
The input will be provided in the following JSON structure:

```
{
  "Damage Type": "<Type of the road crack>",
  "Width": <Width of the crack in meters>,
  "Height": <Height of the crack in meters>
}
```

Output
Your response must be a single JSON object in the following str

```
{
  "Repair Items": [
    {
      "ItemDescription": "Surface Cleaning and Preparation",
      "Quantity": 1,
      "UnitPrice": 250
    },
    ...
  ]
}
```

```

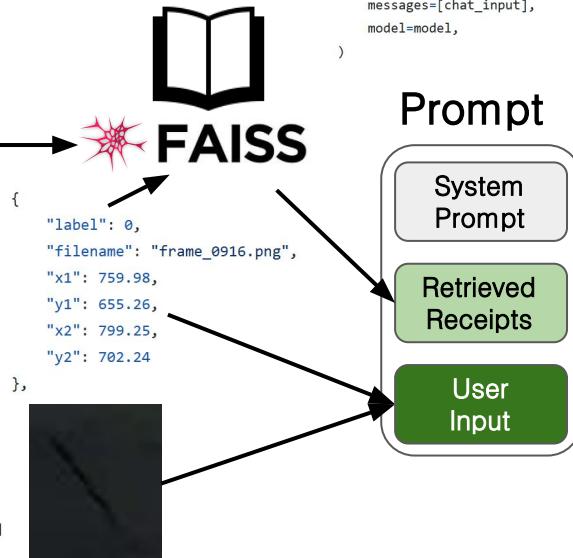
def load_dummy_data():
    """
    Load dummy data from the DUMMY_DATA_FILE, encode them using SBERT,
    and build a FAISS index for similarity search.
    """

    global faiss_index, sentences
    try:
        with open(DUMMY_DATA_FILE, "r") as file:
            dummy_data = json.load(file)

        sentences = [json.dumps(item) for item in dummy_data if isinstance(item, dict)]
        unique_sentences = list(set(sentences))
        embeddings = sbert_model.encode(unique_sentences)

        dimension = embeddings.shape[1]
        faiss_index = faiss.IndexFlatL2(dimension)
        faiss_index.add(embeddings)
    
```

BERT



```

def get_lmm_result(lmm_input, crack_agent_prompt, model):
    """
    Get repair cost estimation from the LMM (OpenAI model).

    :param lmm_input: Dictionary with damage type, dimensions, and cropped image data.
    :param crack_agent_prompt: The system/user prompt to send to LMM.
    :param model: The OpenAI model ID to use for chat completion.
    :return: A dictionary parsed from the LMM's JSON response.
    """

    similar_data = retrieve_similar_data(lmm_input)
    chat_input = {
        "role": "user",
        "content": f"{crack_agent_prompt}\n\nSimilar Data: {json.dumps(similar_data)}\n\nInput: {json.dumps(lmm_input)}"
    }

```

```

chat_completion = client.chat.completions.create(
    messages=[chat_input],
    model=model,
)

```

```

[Request] Request received
Batches: 100%
[Retrieval] Retrieved indices: [92, 62, 44]
HTTP Request: POST http://localhost:9082/v1/chat/completions "HTTP/1.1 200 OK"

```

5. Cooperate with PHI-3.5 LMM



Train Road Damage Detection
YOLOv8 with RDD2022



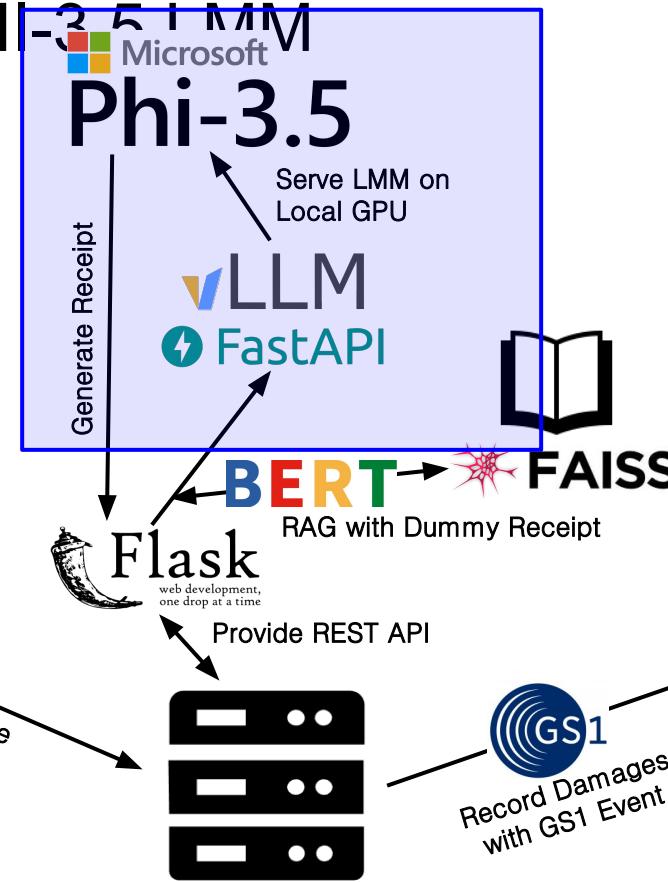
Porting YOLOv8
to XILINX FPGA



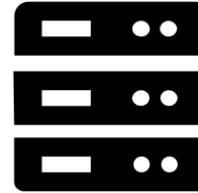
Real-time road
damage detection on
car



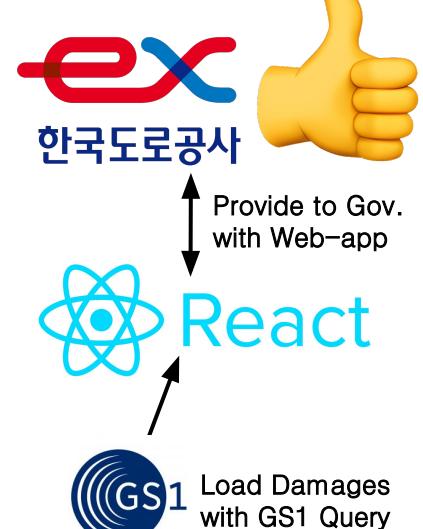
Communicate
with Socket



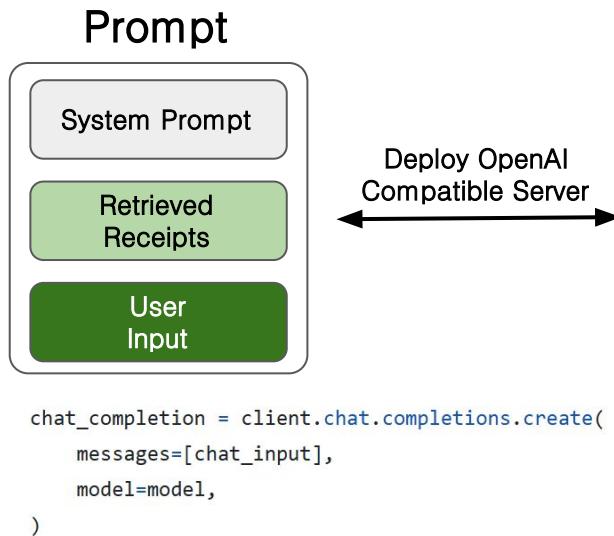
Record Damages
with GS1 Event



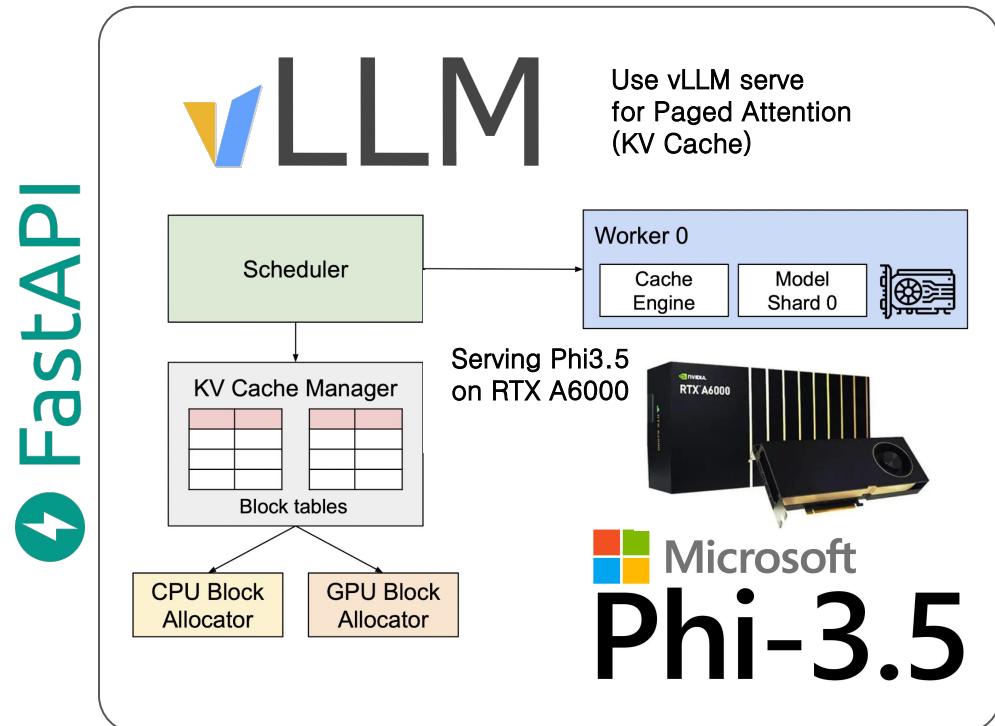
EPCIS



5. Generate with PHI-3.5 LMM



```
vllm serve microsoft/Phi-3.5-vision-instruct --trust-remote-code --max-model-len 8192 --port 8082
```



System Prompt

INFO: ::1:42512 - "POST /v1/chat/completions HTTP/1.1" 200 OK

INFO 12-15 15:31:26 logger.py[7] Received request chatcomplete from user@192.168.0.100:4000. prompt: <user>/<you are a road crack repair cost estimator agent. Your task is to calculate repair quantities and unit prices based on the given crack data and generate a structured JSON response. Your output must strictly follow the JSON format and contain no additional text, comments, or explanation. You can use the input provided in the following JSON structure:</user>`{\n \"Damage Type\": \"Type of the road crack\", \n \"Width\": \"Width of the crack in meters\", \n \"Height\": \"Height of the crack in meters\"}\n` The input will be provided as a single JSON object in the following structure:</user>`{\n \"Repair Items\": [\n {\n \"Item Description\": \"Crack Filling Material (Epoxy/Asphalt)\",\n \"Quantity\": 1,\n \"Unit Price\": 250\n },\n {\n \"Item Description\": \"Equipment Rental (Crack Sealing Machine)\",\n \"Quantity\": 1,\n \"Unit Price\": 120\n },\n {\n \"Item Description\": \"Labor Costs for Repair Team\", \n \"Quantity\": 4,\n \"Unit Price\": 120\n },\n {\n \"Item Description\": \"Post-Repair Inspection and Testing\", \n \"Quantity\": 1,\n \"Unit Price\": 200\n }\n]\n}` Your output **must** be a valid JSON object. Do not include any additional text, explanation, or formatting.</user>2. The 'Repair Items' array must include the six fixed repair items with the exact ItemDescription as provided. **Ans.** Calculate the Quantity and UnitPrice for each repair item based on the following rules:
- **Surface Cleaning and Preparation:** - Quantity: 1, UnitPrice: \$200.
- **Crack Filling Material (Epoxy/Asphalt):** - Quantity: Width * Height, UnitPrice: \$6.
- **Equipment Rental (Crack Sealing Machine):** - Quantity: 4, UnitPrice: \$120.
- **Labor Costs for Repair Team:** - Quantity: 1, UnitPrice: \$300.
- **Post-Repair Inspection and Testing:** - Quantity: 1, UnitPrice: \$200.
- **Traffic Management (Temporary Signs/Barriers):** - Quantity: 1, UnitPrice: \$120.
Example Input:</user>`{\n \"Damage Type\": \"longitudinal crack\", \n \"Width\": 61.38\n}`
Example Output:</user>`{\n \"Repair Items\": [\n {\n \"Item Description\": \"Surface Cleaning and Preparation\", \n \"Quantity\": 1,\n \"Unit Price\": 144.66\n },\n {\n \"Item Description\": \"Crack Filling Material (Epoxy/Asphalt)\",\n \"Quantity\": 44.24,\n \"Unit Price\": 0.32\n },\n {\n \"Item Description\": \"Equipment Rental (Crack Sealing Machine)\",\n \"Quantity\": 3,\n \"Unit Price\": 258\n },\n {\n \"Item Description\": \"Labor Costs for Repair Team\", \n \"Quantity\": 1,\n \"Unit Price\": 417\n },\n {\n \"Item Description\": \"Post-Repair Inspection and Testing\", \n \"Quantity\": 1,\n \"Unit Price\": 143\n },\n {\n \"Item Description\": \"Traffic Management (Signs/Barriers)\",\n \"Quantity\": 1,\n \"Unit Price\": 101\n }\n]\n}`
Any output that is not a valid JSON object will be considered invalid.
Similar Data: [{"Damage Type": "transverse crack", "Width": 58.52, "Height": 74.35, "Repair Items": [{"Item Description": "Surface Cleaning and Preparation", "Quantity": 1, "Unit Price": 394.95}, {"Item Description": "Crack Filling Material (Epoxy/Asphalt)", "Quantity": 372.74, "Unit Price": 0.16}, {"Item Description": "Equipment Rental (Crack Sealing Machine)", "Quantity": 1, "Unit Price": 386}, {"Item Description": "Post-Repair Inspection and Testing", "Quantity": 2, "Unit Price": 247}, {"Item Description": "Labor Costs for Repair Team", "Quantity": 1, "Unit Price": 257}, {"Item Description": "Traffic Management (Signs/Barriers)", "Quantity": 1, "Unit Price": 91}], {"Damage Type": "alligator crack", "Width": 89.28, "Height": 74.76, "Repair Items": [{"Item Description": "Surface Cleaning and Preparation", "Quantity": 1, "Unit Price": 598.14}, {"Item Description": "Crack Filling Material (Epoxy/Asphalt)", "Quantity": 91.83, "Unit Price": 0.06}, {"Item Description": "Equipment Rental (Crack Sealing Machine)", "Quantity": 1, "Unit Price": 419}, {"Item Description": "Post-Repair Inspection and Testing", "Quantity": 1, "Unit Price": 247}, {"Item Description": "Labor Costs for Repair Team", "Quantity": 4, "Unit Price": 247}, {"Item Description": "Traffic Management (Signs/Barriers)", "Quantity": 1, "Unit Price": 101}]}]

Input Json

Retrieved Receipts

Input Image

Prefill Phase

Decoding Phase

No KV Cache



Train Road Damage Detection
YOLOv8 with RDD2022



Porting YOLOv8
to XILINX FPGA



Real-time road
damage detection on
car



Recording Events with GS1 EPCISv2



Phi-3.5

Serve LMM on
Local GPU

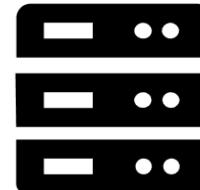


BERT

RAG with Dummy Receipt



Provide REST API



RAG with Dummy Receipt



Provide to Gov.
with Web-app



Load Damages
with GS1 Query



Record Damages
with GS1 Event



Store GS1 Data

6. Recording Events with GS1 EPCISv2



```
def send_to_epcis(parsed_result, lmm_input, boxed_image_path):
    """
    Send event data to the EPCIS server.

    :param parsed_result: The final dictionary containing cost info and damage details.
    :param lmm_input: The dictionary with dimension and damage type.
    :param boxed_image_path: Path to the boxed image file.
    :return: The text response from the EPCIS server.
    :raises Exception: If the EPCIS server returns a non-success status code.
    """
    epcis_post_data = {
        "@context": [
            "https://ref.gs1.org/standards/epcis/2.0.0/epcis-context.jsonld",
            {
                ...
            }
        ],
        headers = {
            "Content-Type": "application/json",
            "GS1-EPCIS-Version": "2.0.0",
            "GS1-CBV-Version": "2.0.0"
        }
        epcis_server_url = "http://192.168.4.7:8090/epcis/v2/events"
        response = requests.post(epcis_server_url, headers=headers, json=epcis_post_data)
        if response.status_code not in [200, 201, 202]:
            raise Exception(
                f"Failed to send POST request to EPCIS server "
                f"(Status {response.status_code}): {response.text}"
            )
        return response.text
    }
```

POST /epcis/v2/events
GS1



GET /epcis/v2/events
GS1



```
redis | 1:C 13 Dec 2024 12:33:17.249 * 000000000000
redis | 1:C 13 Dec 2024 12:33:17.249 * Redis version
redis | 1:C 13 Dec 2024 12:33:17.249 * Configuration
redis | 1:M 13 Dec 2024 12:33:17.249 * monotonic cl
redis | 1:M 13 Dec 2024 12:33:17.251 * Running mode
redis | 1:M 13 Dec 2024 12:33:17.251 * Server initiali
redis | 1:M 13 Dec 2024 12:33:17.251 * Reading RDB
redis | 1:M 13 Dec 2024 12:33:17.251 * Loading RDB
redis | 1:M 13 Dec 2024 12:33:17.251 * RDB age 2651
redis | 1:M 13 Dec 2024 12:33:17.251 * RDB memory u
redis | 1:M 13 Dec 2024 12:33:17.251 * RDB is base
redis | 1:M 13 Dec 2024 12:33:17.251 * Done loading
redis | 1:M 13 Dec 2024 12:33:17.251 * DB loaded fr
redis | 1:M 13 Dec 2024 12:33:17.275 * DB loaded fr
redis | 1:M 13 Dec 2024 12:33:17.275 * DB loaded fr
redis | 1:M 13 Dec 2024 12:33:17.275 * Opening AOF
redis | 1:M 13 Dec 2024 12:33:17.275 * Ready to acc
redis | 1:M 13 Dec 2024 12:34:18.023 * 10000 change
redis | 1:M 13 Dec 2024 12:34:18.024 * Background s
redis | 21:C 13 Dec 2024 12:34:18.034 * DB saved on
redis | 21:C 13 Dec 2024 12:34:18.034 * Fork CoW fo
redis | 1:M 13 Dec 2024 12:34:18.125 * Background s
redis | 1:M 13 Dec 2024 13:23:49.150 * 100 changes
redis | 1:M 13 Dec 2024 13:23:49.151 * Background s
redis | 22:C 13 Dec 2024 13:23:49.162 * DB saved on
redis | 22:C 13 Dec 2024 13:23:49.162 * Fork CoW fo
redis | 1:M 13 Dec 2024 13:23:49.252 * Background s
epcis | 2024-12-13T13:29:51.335Z 3 Events returned
epcis | 2024-12-13T13:29:51.380Z 3 Events returned
epcis | 2024-12-13T13:31:10.175Z 6 Events returned
epcis | 2024-12-13T13:31:10.223Z 6 Events returned
```

```
const App = () => {
  const [events, setEvents] = useState([]);
  const [selectedEvent, setSelectedEvent] = useState(null);
  const [open, setOpen] = useState(false);

  useEffect(() => {
    axios
      .get("http://localhost:8090/epcis/v2/events", {
        headers: {
          "Content-Type": "application/json",
          "GS1-EPCIS-Version": "2.0.0",
          "GS1-CBV-Version": "2.0.0",
        },
      })
      .then((response) => {
        const eventList = response.data?.epcisBody?.queryResults?.resultBody?.eventList || [];

        const cleanedData = eventList.map((event) => ({
          eventTime: event.eventTime || "N/A",
          geoLocation: event.readPoint?.id || "Unknown",
        }));
      })
      .catch((error) => {
        console.error(error);
      });
  }, []);

  const handleEventClick = (event) => {
    setSelectedEvent(event);
    setOpen(true);
  };

  const handleClose = () => {
    setSelectedEvent(null);
    setOpen(false);
  };

  return (
    <div>
      <h1>EPCIS Events</h1>
      <p>Fetching data from http://localhost:8090/epcis/v2/events...</p>
      <p>Filtering unprocessed events...</p>
      <table border="1">
        <thead>
          <tr>
            <th>Event Time</th>
            <th>EPC List</th>
            <th>Geo Location</th>
            <th>Damage Type</th>
            <th>Repair Cost</th>
            <th>Repair Items</th>
          </tr>
        </thead>
        <tbody>
          <tr>
            <td>2024-12-13T13:29:51.335Z</td>
            <td>[{"urn:epc:id:road:0012345"]</td>
            <td>urn:epc:id:geo:36.365365,127.358708</td>
            <td>longitudinal crack</td>
            <td>1892.1544000000001</td>
            <td>[{"ItemDescription": "Surface Cleaning and Preparation", "Quantity": 1, "UnitPrice": 144.66, "TotalPrice": 144.66}</td>
          </tr>
        </tbody>
      </table>
    </div>
  );
}

export default App;
```

6. Recording Events with EPCIS 2.0

EPCIS 2.0: A web language for supply chain events & interop

What? When? Where? Why? How?

Source: evrythng.com

```
"epcList":  
["urn:epc:id:road:0012345"]
```

```
"eventTime":  
"2024-12-13T13:31:28.587875Z"
```

```
"readPoint":  
"urn:epc:id:geo:urn:epc:id:geo:36.365365,127.358708"
```

```
"bizStep":  
"repairing",  
"disposition":  
"in_progress"
```

```
"ilmd": {  
    "Damage Type": "longitudinal crack",  
    "Repair Cost": 1892.1544000000001,  
    "Repair Items": [  
        {  
            "ItemDescription": "Surface Cleaning and Preparation",  
            "Quantity": 1,  
            "UnitPrice": 144.66,  
            "TotalPrice": 144.66  
        },  
        {  
            "ItemDescription": "Crack Filling Material (Epoxy/Asphalt)",  
            "Quantity": 2225.24,  
            "UnitPrice": 0.06,  
            "TotalPrice": 133.5144  
        },  
        {  
            "ItemDescription": "Labor Costs for Repair Team",  
            "Quantity": 2,  
            "UnitPrice": 248,  
            "TotalPrice": 496  
        }  
    ],  
    "Dimensions": {  
        "Width": 47.32000000000005,  
        "Height": 48.98000000000002  
    }  
}
```

```
{  
    "ItemDescription": "Equipment Rental (Crack Sealing Machine)",  
    "Quantity": 1,  
    "UnitPrice": 872.32,  
    "TotalPrice": 872.32  
},  
{  
    "ItemDescription": "Post-Repair Inspection and Testing",  
    "Quantity": 1,  
    "UnitPrice": 144.66,  
    "TotalPrice": 144.66  
},  
{  
    "ItemDescription": "Traffic Management (Signs/Barriers)",  
    "Quantity": 1,  
    "UnitPrice": 101,  
    "TotalPrice": 101  
}  
],  
"Dimensions": {  
    "Width": 47.32000000000005,  
    "Height": 48.98000000000002  
}  
}
```



Train Road Damage Detection
YOLOv8 with RDD2022



Porting YOLOv8
to XILINX FPGA



Real-time road
damage detection on
car



Communicate
with Socket

Provide Records to Gov. by React-web

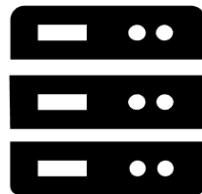
Microsoft Phi-3.5

Serve LMM on
Local GPU

LLM
FastAPI

BERT → FAISS
RAG with Dummy Receipt

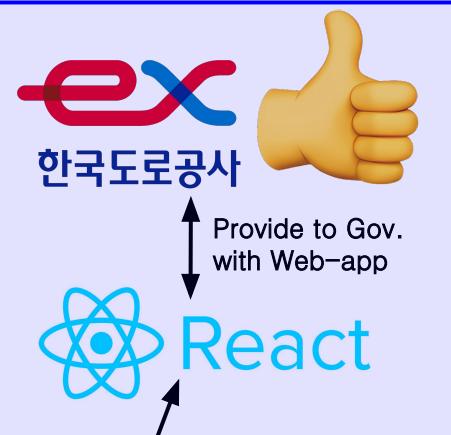
Flask
web development,
one drop at a time



Provide REST API



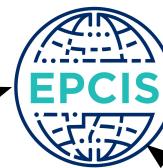
Record Damages
with GS1 Event



Provide to Gov.
with Web-app



GS1 Load Damages
with GS1 Query

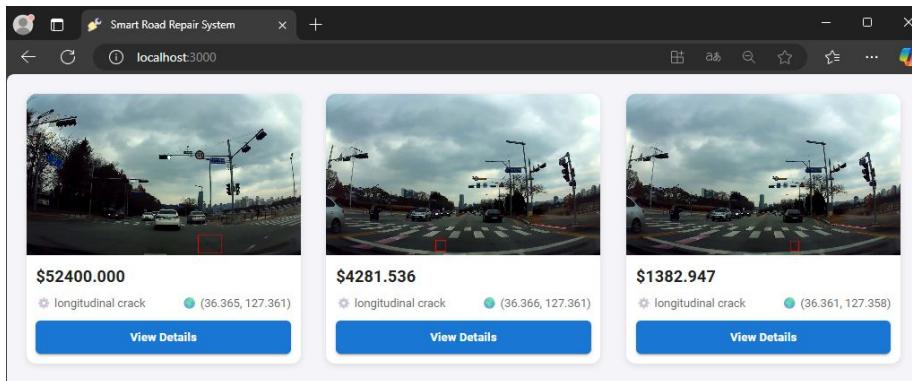


EPCIS

Store GS1 Data
MongoDB
redis

7. Provide Records to Gov. by React-web

For Government, Provide Total cost, size, location, label by React Webapp



The screenshot shows a "Receipt" modal window overlaid on a web browser window for "Smart Road Repair System" on "localhost:3000". The receipt details the breakdown of the repair costs shown in the dashboard. It includes a header with the total amount (\$52400.000), a section for the first task (\$4281.536), and a detailed breakdown of the third task (\$1382.947). The breakdown lists various repair components with their quantities and unit prices.

Description	Quantity	Unit Price	Total Price
Surface Cleaning and Preparation	1.00	\$250.00	\$250.00
Crack Filling Material (Epoxy/Asphalt)	11.26	\$256.00	\$2881.54
Labor Costs for Repair Team	4.00	\$120.00	\$480.00
Equipment Rental (Crack Sealing Machine)	1.00	\$350.00	\$350.00
Post-Repair Inspection and Testing	1.00	\$200.00	\$200.00
Traffic Management (Signs/Barriers)	1.00	\$120.00	\$120.00

System Demos

add demos with videos as input

Q&A