Efficient Road Repairs System

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

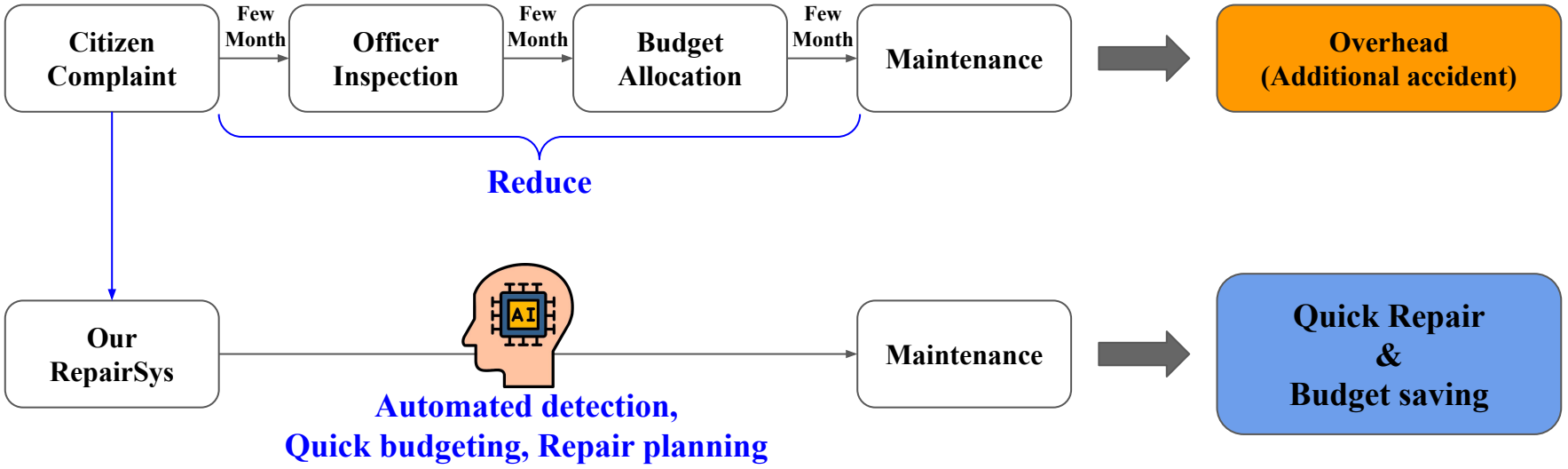
Team #1, Final project presentation

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

1. SYSTEM OVERVIEW

Train Road Damage Detection
YOLOv8 with RDD2022

ultralytics YOLO

Porting YOLOv8
to XILINX FPGA

XILINX

Real-time road
damage detection on
car

FPGA

Communicate
with Socket

Microsoft Phi-3.5

Serve LMM on Local GPU

Generate Receipt

LLM

FastAPI

BERT

FAISS

RAG with Dummy Receipt

Flask
web development,
one drop at a time

Provide REST API

GS1

Record Damages
with GS1 Event

EPCIS

Store GS1 Data

MongoDB.

redis

GS1

Load Damages
with GS1 Query

React

Provide to Gov.
with Web-app

한국도로공사

2. Real-time Road Damage Detection on FPGA

Train Road Damage Detection
YOLOv8 with RDD2022

ultralytics YOLO

Porting YOLOv8
to XILINX FPGA

XILINX

Real-time road
damage detection on
car

FPGA

Communicate
with Socket

Microsoft
Phi-3.5

Serve LMM on
Local GPU

Generate Receipt

LLM
FastAPI

BERT

FAISS

RAG with Dummy Receipt

Flask
web development,
one drop at a time

Provide REST API

GS1
Record Damages
with GS1 Event

EPCIS

Store GS1 Data

MongoDB.

redis

한국도로공사

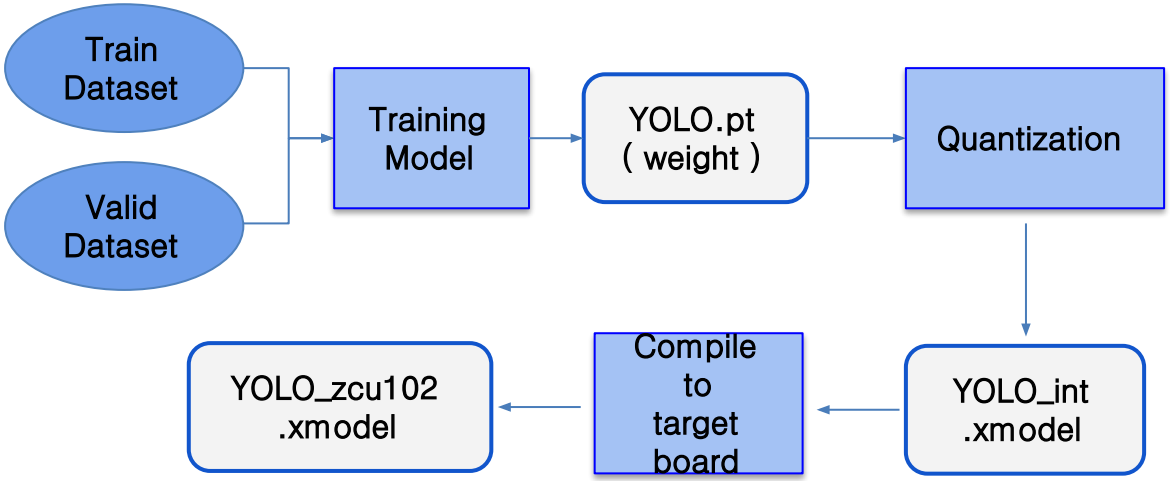
Provide to Gov.
with Web-app

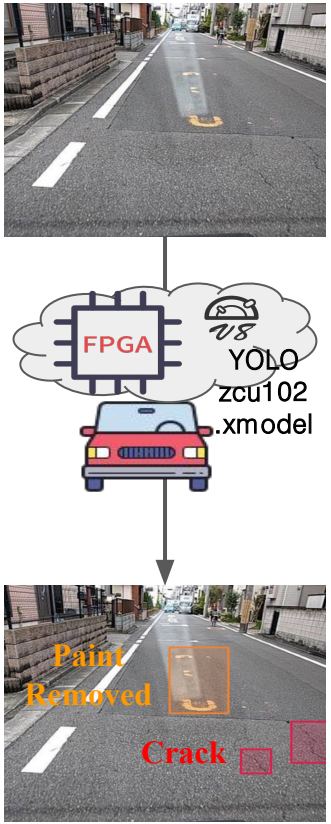
React

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

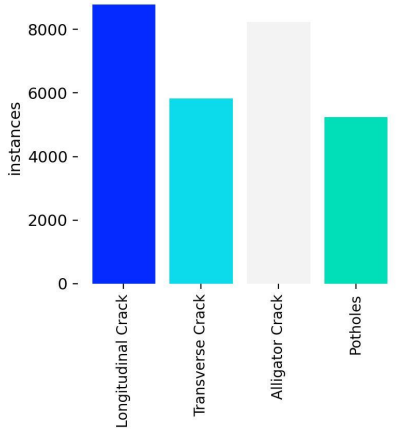
2.1 YOLOv8-Enabled Road Damage Detection

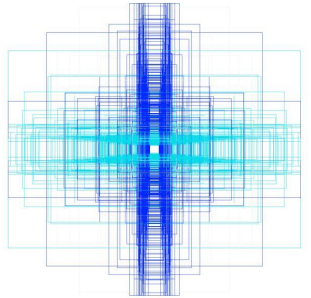
Dataset: Road Damage Detection Challenge 2022 dataset (RDD 2022)

Model trained: YOLOv8m

```
├── RDD2022_all_countries
│   ├── China_Drone
│   │   └── train
│   │       ├── annotations
│   │       │   └── xmls
│   │       ├── images
│   │       └── labels # Created after prepare dataset process
│   ├── China_MotorBike
│   │   └── ...
│   ├── Czech
│   │   └── ...
│   ├── India
│   │   └── ...
│   ├── Japan
│   │   └── ...
│   ├── Norway
│   │   └── ...
│   └── United_States
│       └── ...
```

Samples utilized
Asian country division:
China Drone & China MotorBike & India & Japan

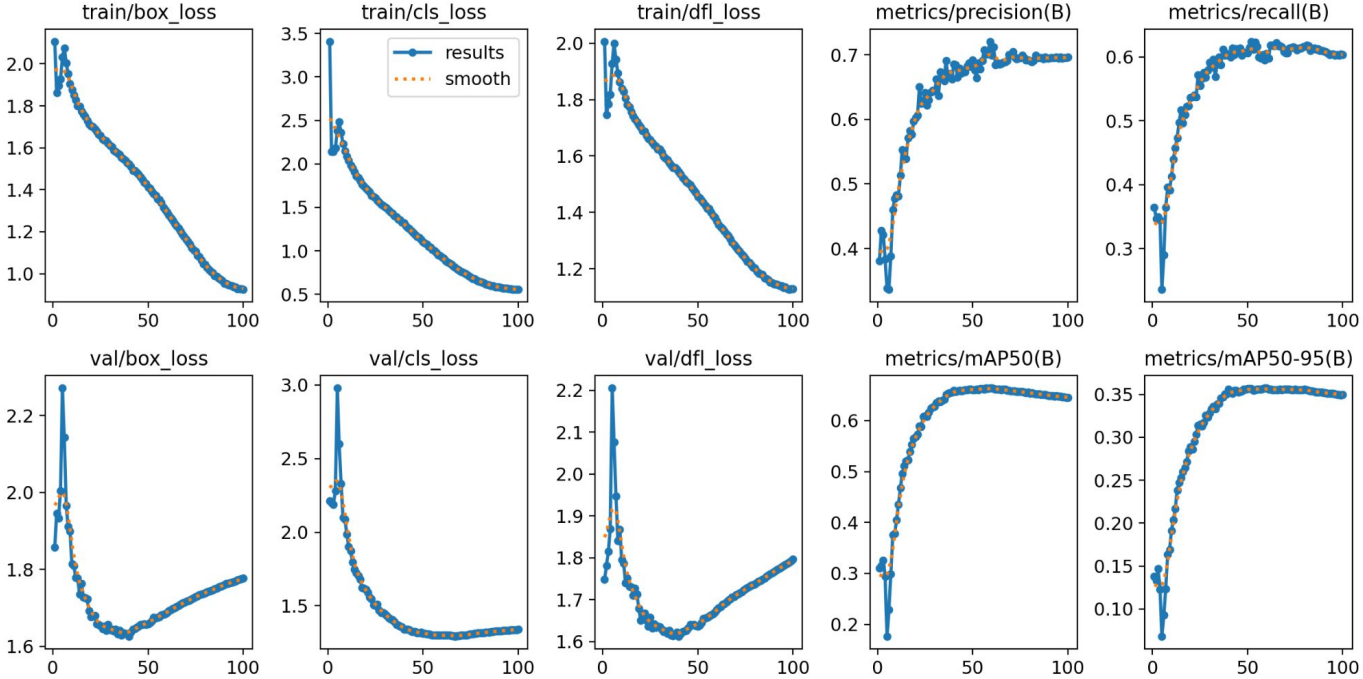




- Dataset conversion from PascalVOC to YOLOv8 format.
- Train and val dataset split: 9:1
- Training & Evaluating YOLOv8 ultralytics
- Image cropping on detected objects

2.1 YOLOv8-Enabled Road Damage Detection

YOLOv8m Model Performance

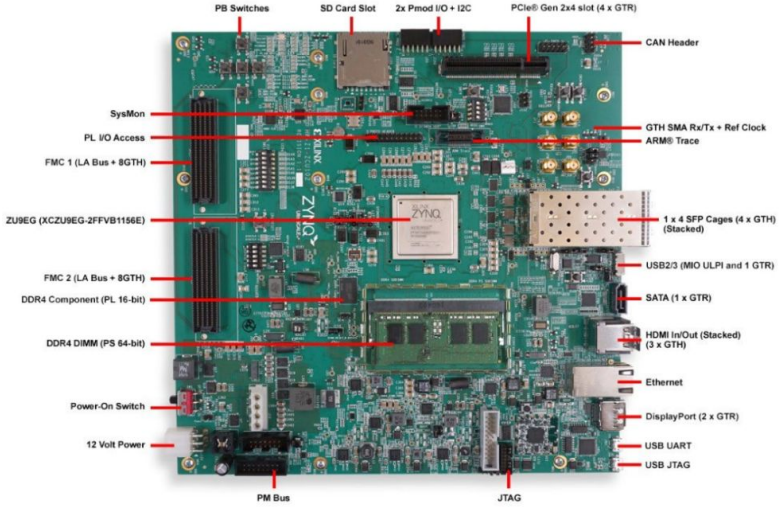


2.1 YOLOv8-Enabled Road Damage Detection

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

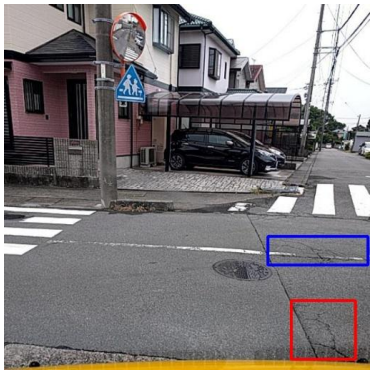
- Quantization
  - Reduce Memory Usage, Computation on FPGAs
  - Performing INT 8-bit quantization
- Compilation
  - Compile to the architecture of the FPGA (ZCU102)
  - Optimized for DPU utilization
  - Change activation function for DPU compatibility (Silu -> Hardswish)
  - YOLO_post_process

    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

```cpp
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
            {
                detect_output_tensors.emplace_back(output_tensor_buffers[j]);
            }
        }
```



	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**

```cpp
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;
}
```

Image Data

```cpp
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]) + " " ; }
```

Text Data

```python
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)
```

3. REST API Server

Train Road Damage Detection
YOLOv8 with RDD2022

ultralytics YOLO

Porting YOLOv8
to XILINX FPGA

XILINX

Real-time road
damage detection on
car

FPGA

Communicate
with Socket

Microsoft
Phi-3.5

Serve LMM on
Local GPU

Generate Receipt

LLM
FastAPI

BERT

FAISS

RAG with Dummy Receipt

Flask
web development,
one drop at a time

Provide REST API

GS1
Record Damages
with GS1 Event

EPCIS

Store GS1 Data

MongoDB.
redis

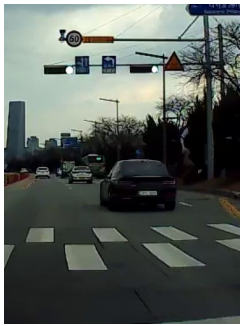
GS1
Load Damages
with GS1 Query

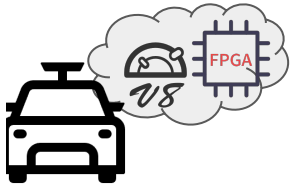
React

Provide to Gov.
with Web-app

한국도로공사

3. REST API Server





POST /process
To get img & json

Flask
web development,
one drop at a time

Microsoft
Phi-3.5

Post-processing
Generated Receipt

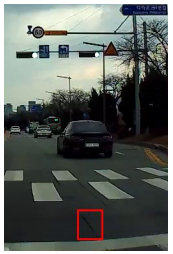
Send to LLMEngine with RAG & Img

FAISS

BERT

React

GET /static/images
To Serve Boxed Img



GS1

EPCIS

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

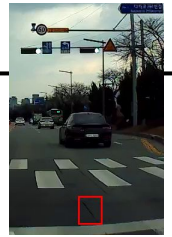
BERT
Download Pretrained bert-base-nli-mean-tokens

FAISS
Index dummy receipts

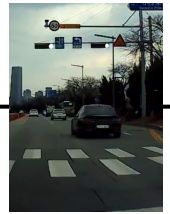
Flask
web development, one drop at a time
Run Server

Microsoft Phi-3.5
Connect with Phi-3.5 vLLM server

GET /static/images
To Serve Boxed Img



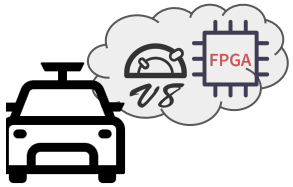
POST /process
To get img & json




```
{
    "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 product
ion 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 -
172.17.0.1 - - [13/Dec/2024 13:31:10] "GET /static/images/boxed_20241213_123346.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



React

```
@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.
    """
```

Flask
web development,
one drop at a time

```
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.
    """
```

GS1

EPCIS

4. RAG Pipeline

Train Road Damage Detection
YOLOv8 with RDD2022

ultralytics YOLO

Porting YOLOv8
to XILINX FPGA

XILINX

Real-time road
damage detection on
car

FPGA

Communicate
with Socket

Microsoft
Phi-3.5

Serve LMM on
Local GPU

Generate Receipt

LLM
FastAPI

BERT → FAISS
RAG with Dummy Receipt

Flask
web development,
one drop at a time

Provide REST API

GS1
Record Damages
with GS1 Event

EPCIS

Store GS1 Data

MongoDB.
redis

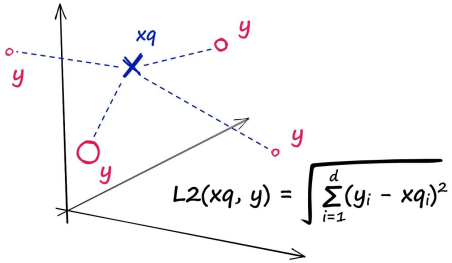
한국도로공사

Provide to Gov.
with Web-app

React

GS1 Load Damages
with GS1 Query

4. RAG Pipeline




```
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
    },
```

```
{
    "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": 39.2699999999998,
        "Height": 46.9800000000002
    }
}
```

x 10000
Receipt Database

BERT
Indexing with sBERT + IndexFlatL2

FAISS

Top 3 R...

Receipt_0
Receipt_1
Receipt_2

Retrieve Similar Receipts

Prompt

System Prompt
Retrieved Receipts
User Input

Construct Prompt

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

Yolo detection data

FPGA

Image file

Generate Receipt

Microsoft Phi-3.5

4. RAG Pipeline

```python
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 → FAISS

```python
def retrieve_similar_data(input_data):
    """
    Retrieve similar data entries from the FAISS index given an input object.

    :param input_data: Dictionary representing the input for similarity retrieval.
    :return: A list of similar data items.
    """
    if faiss_index is None:
        return []
    try:
        input_sentence = json.dumps(input_data)
        input_embedding = sbert_model.encode([input_sentence])
        D, I = faiss_index.search(input_embedding, k=3)
        log_retrieval(f"Retrieved indices: {I[0].tolist()}")
        similar_data = [json.loads(sentences[i]) for i in I[0] if i < len(sentences)]
        return similar_data
```

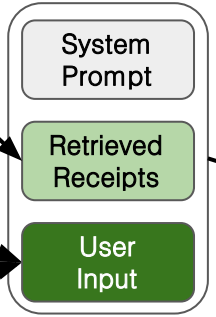
```python
def get_lmm_result(lmm_input, crack_agent_prompt, model):
    """
    Get repair cost estimation from the LMM (OpenAI model).

    :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%|                                          | 1/1 [00:00<0
[Retrieval] Retrieved indices: [92, 62, 44]
HTTP Request: POST http://localhost:8082/v1/chat/completions "HTTP/1.1 200 OK"
```

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



Generate

Microsoft Phi-3.5

5. Generate with PHI-3.5 LMM

Train Road Damage Detection
YOLOv8 with RDD2022

ultralytics YOLO

Porting YOLOv8
to XILINX FPGA

XILINX

Real-time road
damage detection on
car

FPGA

Communicate
with Socket

Microsoft Phi-3.5

Serve LMM on
Local GPU

vLLM

FastAPI

Generate Receipt

BERT

FAISS

RAG with Dummy Receipt

Flask
web development,
one drop at a time

Provide REST API

GS1

Record Damages
with GS1 Event

EPCIS

Store GS1 Data

MongoDB.

redis

한국도로공사

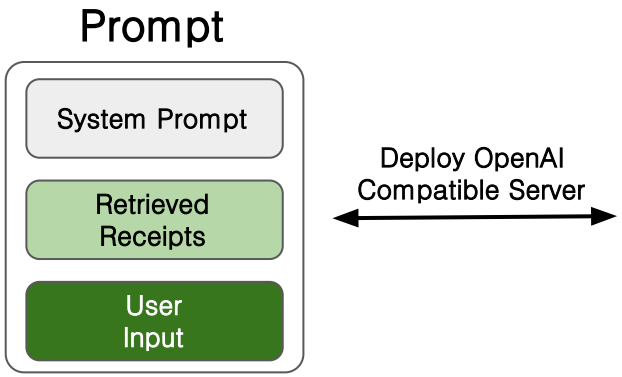
Provide to Gov.
with Web-app

React

GS1 Load Damages
with GS1 Query

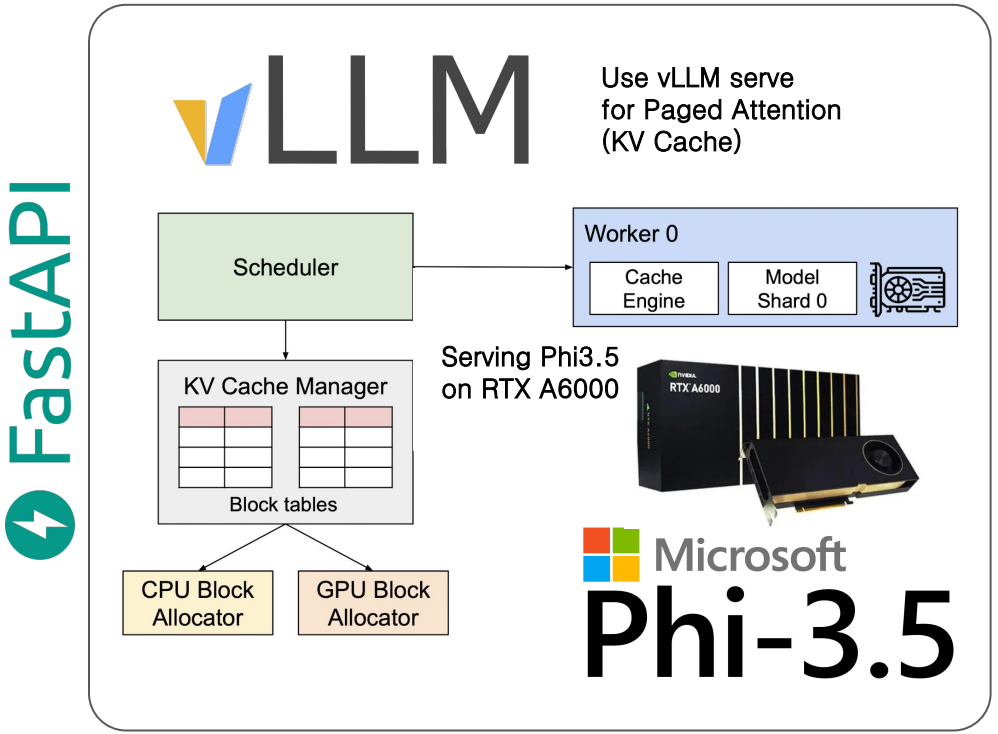
5. Generate with PHI-3.5 LMM

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

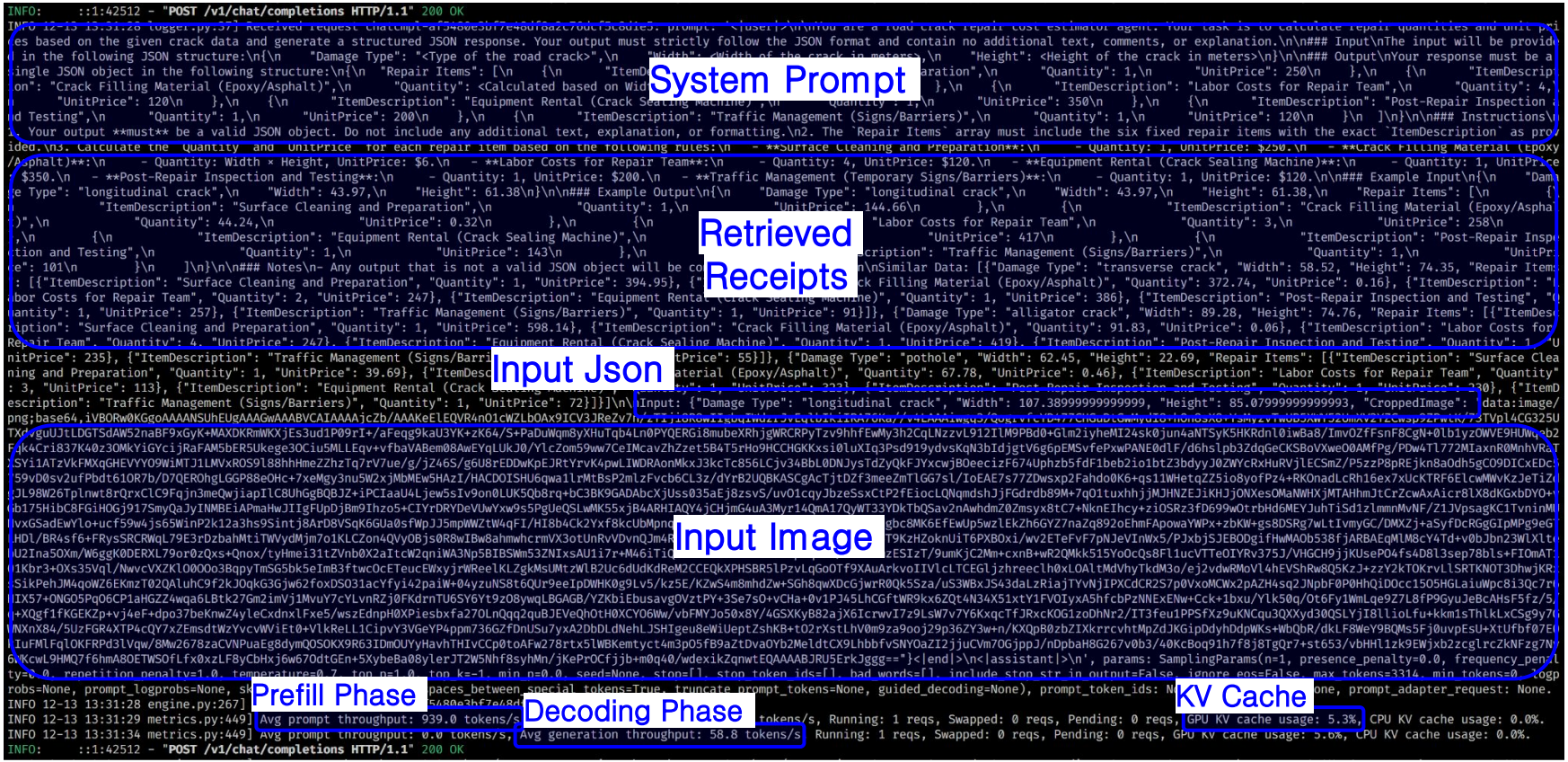


Deploy OpenAI Compatible Server

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



5. Generate with PHI-3.5 LMM



6. Recording Events with GS1 EPCISv2

Train Road Damage Detection
YOLOv8 with RDD2022

ultralytics YOLO

Porting YOLOv8
to XILINX FPGA

XILINX

Real-time road
damage detection on
car

FPGA

Communicate
with Socket

Microsoft
Phi-3.5

Serve LMM on
Local GPU

Generate Receipt

LLM
FastAPI

BERT

FAISS

RAG with Dummy Receipt

Flask
web development,
one drop at a time

Provide REST API

한국도로공사

Provide to Gov.
with Web-app

React

GS1 Load Damages
with GS1 Query

EPCIS

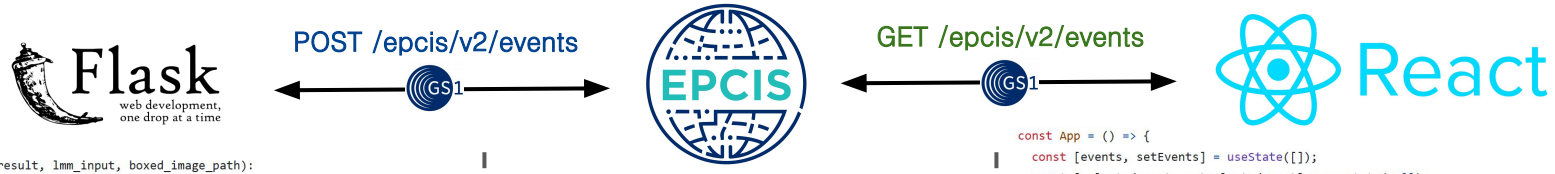
GS1 Record Damages
with GS1 Event

Store GS1 Data

MongoDB.

redis

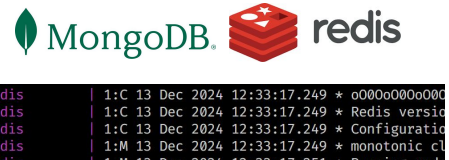
6. Recording Events with GS1 EPCISv2




```python
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
```




```
redis        | 1:C 13 Dec 2024 12:33:17.249 * oO0OoO0OoO0O
redis        | 1:C 13 Dec 2024 12:33:17.249 * Redis versio
redis        | 1:C 13 Dec 2024 12:33:17.249 * Configuratio
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 initi
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
```

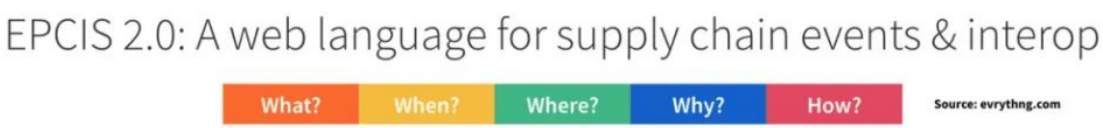
```javascript
const App = () => {
  const [events, setEvents] = useState([]);
  const [selectedEvent, setSelectedEvent] = useState(null);
  const [open, setOpen] = useState(false);

  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",
```

```
Fetching data from http://localhost:8090/epcis/v2/events...
Filtering unprocessed events...
{
  "Event Time": "2024-12-13T13:31:28.587875Z",
  "EPC List": [
    "urn:epc:id:road:0012345"
  ],
  "Geo Location": "urn:epc:id:geo:36.365365,127.358708",
  "Damage Type": "longitudinal crack",
  "Repair Cost": 1892.1544000000001,
  "Repair Items": [
    {
      "ItemDescription": "Surface Cleaning and Preparation",
      "Quantity": 1,
      "UnitPrice": 144.66,
      "TotalPrice": 144.66
    },
```

6. Recording Events with GS1 EPCISv2




```
"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,
                                  lPrice": 133.5144
    },
    {
      "ItemDescription": "Labor Costs for Repair Team",
      "Quantity": 2,
      "UnitPrice": 248,
      "TotalPrice": 496
    },
    {
      "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
  }
}
}
```

7. Provide Records to Gov. by React-web

Train Road Damage Detection
YOLOv8 with RDD2022

ultralytics YOLO

Porting YOLOv8
to XILINX FPGA

XILINX

Real-time road
damage detection on
car

FPGA

Communicate
with Socket

Microsoft Phi-3.5

Serve LMM on
Local GPU

Generate Receipt

LLM
FastAPI

BERT

FAISS

RAG with Dummy Receipt

Flask
web development,
one drop at a time

Provide REST API

GS1
Record Damages
with GS1 Event

EPCIS

Store GS1 Data

MongoDB.
redis

GS1 Load Damages
with GS1 Query

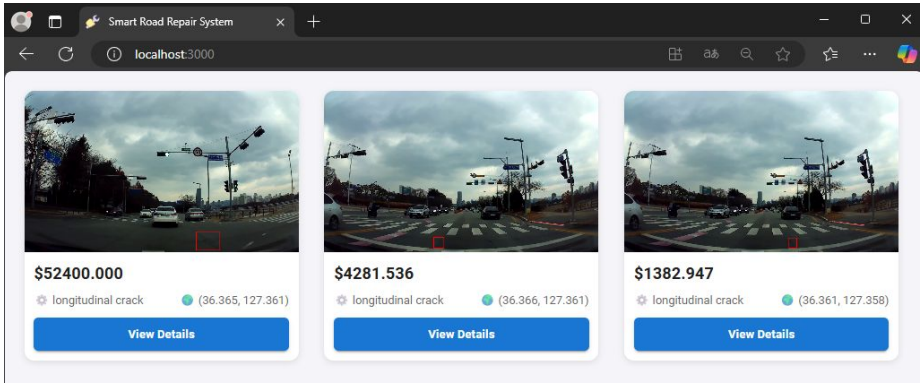
한국도로공사

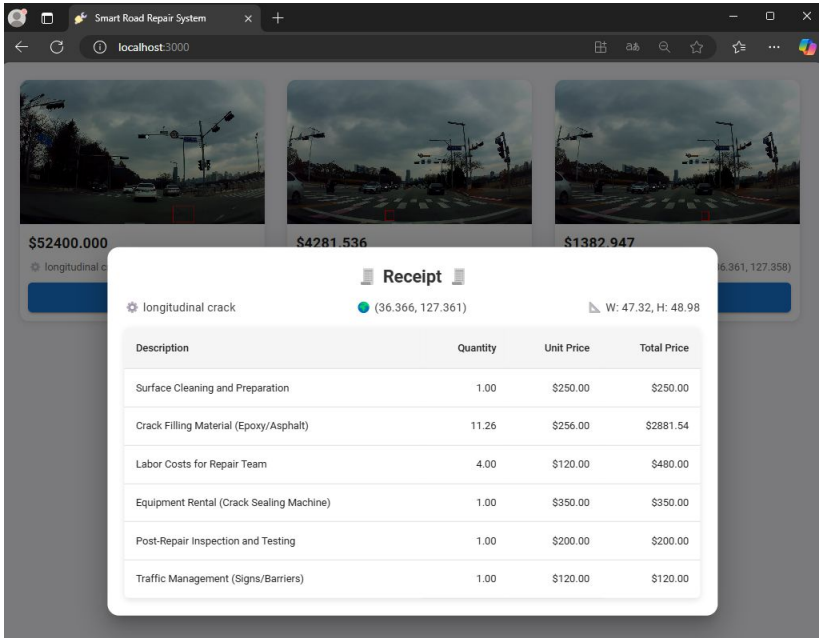
Provide to Gov.
with Web-app

React

7. Provide Records to Gov. by React-web

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





System Demos

https://github.com/SeungjaeLim/Efficient-Road-Repairs-System/blob/main/Demo.mp4

Q&A