

GOOD MORNING!

早上好!

안녕하세요!

PROJECT INTRODUCTION

HOW TO WORK TOGETHER

- Participate, Participate, Participate!!!
- No long emails or Kakaotalk, prefer face to face
- Be open to suggestions and idea
- Be proactive, take initiative
- HOW is as important as WHAT

DAY I (DONE)

- Welcome
- Project Introduction
- Introduction to Project Development Process
- Business Requirement Development
- System Requirement Development
- System(High Level) Design
- Time Management

DAY 2

- YOLOv8 기반 데이터 수집/학습/deploy (Security Alert)
 - 감시용 데이터 수집(bus, truck, tank 등)
 - 감시용 데이터 라벨링
 - YOLOv8 기반 학습
 - YOLOv8 Object Detection
- Porting to ROS
 - Create Security Alert Node
 - Generate Topics to send image and Obj. Det. results
 - Create Subscriber node and display image and print data from the Topic

DAY 2

- Flask 를 이용한 웹 서버 구축 (System Monitor)
 - Flask/HTML Intro
 - Deploy YOLOv8 Obj. Det results to web
 - Log in 기능 구현
 - Sysmon 웹기능 구현
 - 알람 기능 구현
- Porting to ROS
 - Create Sysmon Node
 - Receive Image/Data Topic from Security Alert Node and display on the SysMon webpage

DAY 3

- SQLite3를 이용한 데이터베이스 구축 및 연동 (System Monitor)
 - SQLite3 기본 기능 구현
 - DB 기능 구축
 - 알람이 울리는 경우 DB에 저장하는 기능 구현
 - 저장된 내용 검색하는 기능 구현
- Porting to ROS
 - Update Sysmon Node code
 - Update the database with received Obj. Det. Data from Security Alert Node
 - Display the content of DB on Security Monitor web page

프로젝트 RULE

80/20 → 20/80

TEAMWORK AND PROJECT MANAGEMENT



프로젝트 RULE NUMBER ONE!!!

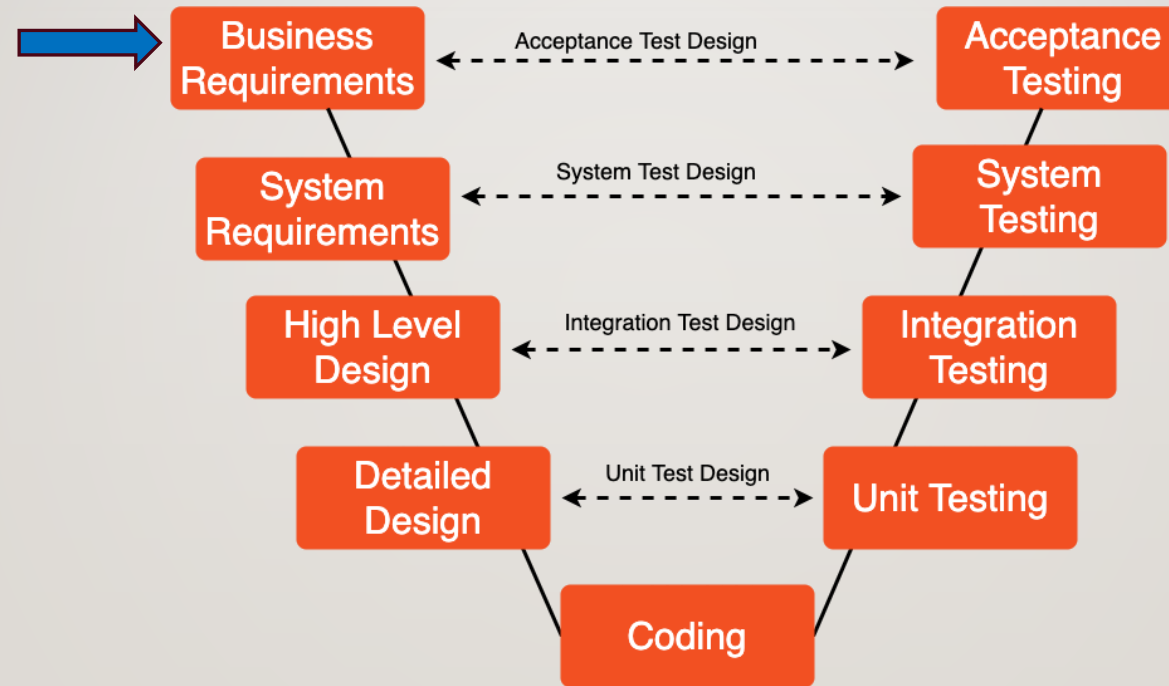
Have Fun Fun Fun!



PROJECT DEVELOPMENT
IS A PROCESS



SW DEVELOPMENT PROCESS



SDLC - V Model - notepub.io

BRAINSTORMING RULES

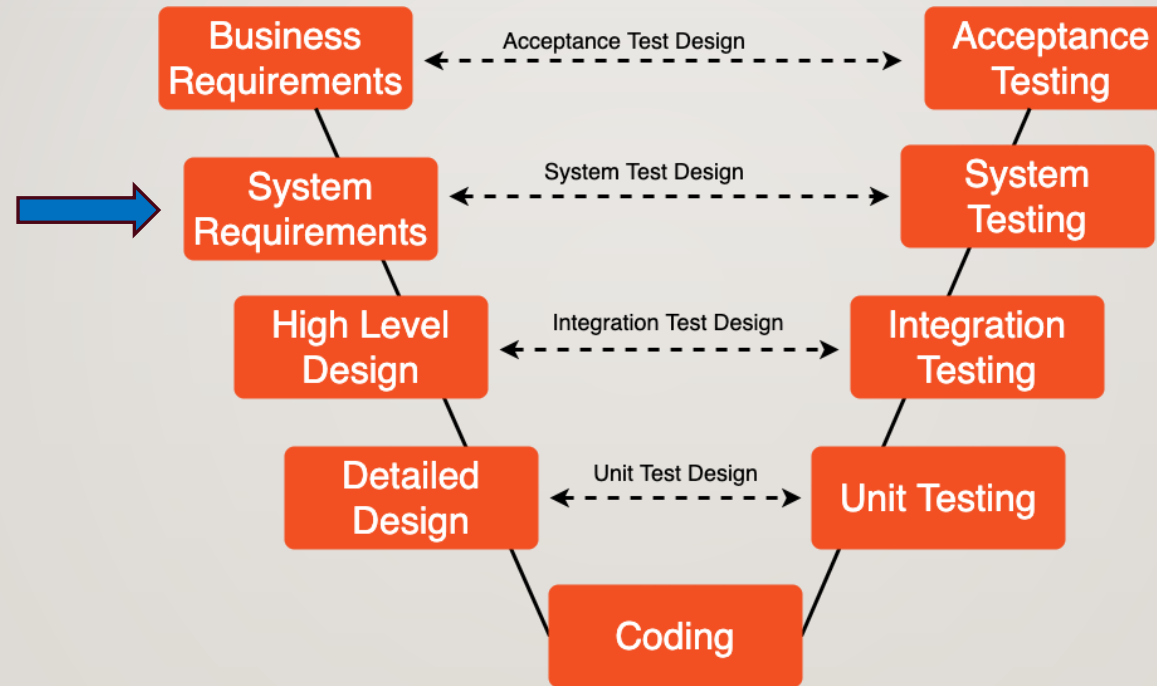
- Every input is good input
- Do not critique inputs only seek to understand
- Organize inputs into logical groupings
- Sequence or show relationships as needed
- Use Posted Notes on Flip Chart



TEAM EXERCISE I

Brainstorm Business Requirement for the project and write business requirement statement

SW DEVELOPMENT PROCESS



SDLC - V Model - notepub.io

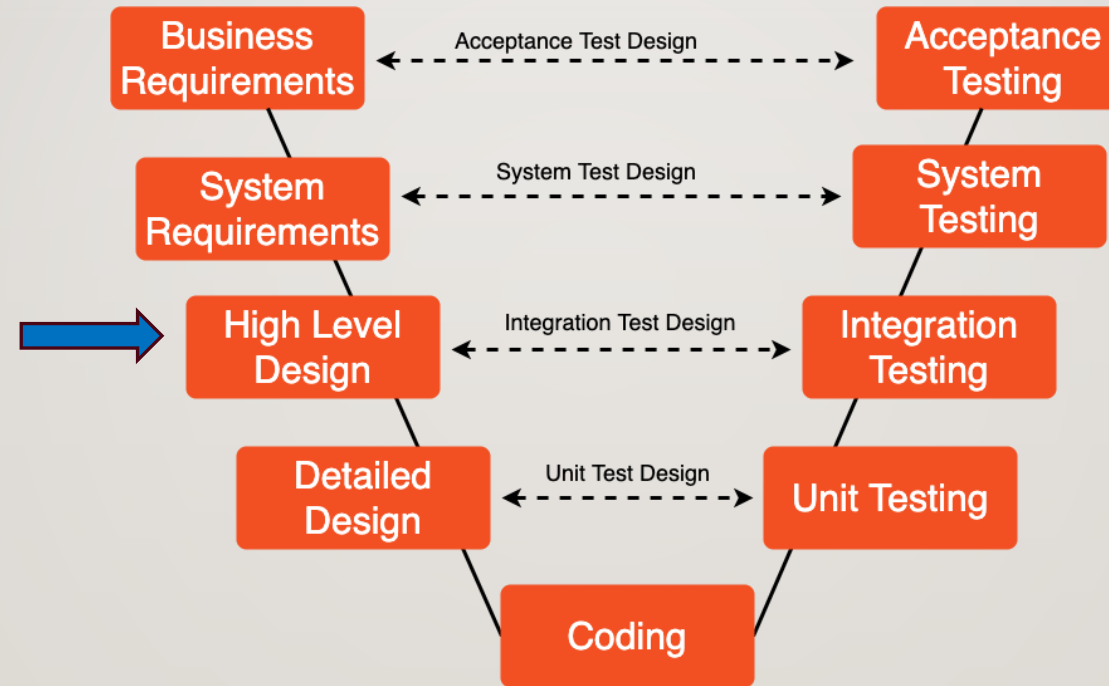
TEAM EXERCISE 2

Brainstorm System Requirement for the project and document

TEAM EXERCISE 3

System Environment and Development Environment Setup

SW DEVELOPMENT PROCESS



SDLC - V Model - notepub.io

KEY FUNCTIONS (MODULES) TO DEVELOP

- Security Alert
 - Object Detection
- System Monitor
 - Display Security Camera and info
 - Display AMR Camera and info
 - Store, display, and report Alerts
- AMR Controller
 - Movement (SLAM)
 - Target Acquisition (Obj. Det.) and Tracking

TEAM EXERCISE 4

Create System Design using Process Flow Diagram.

Use the posted notes and flipchart as needed

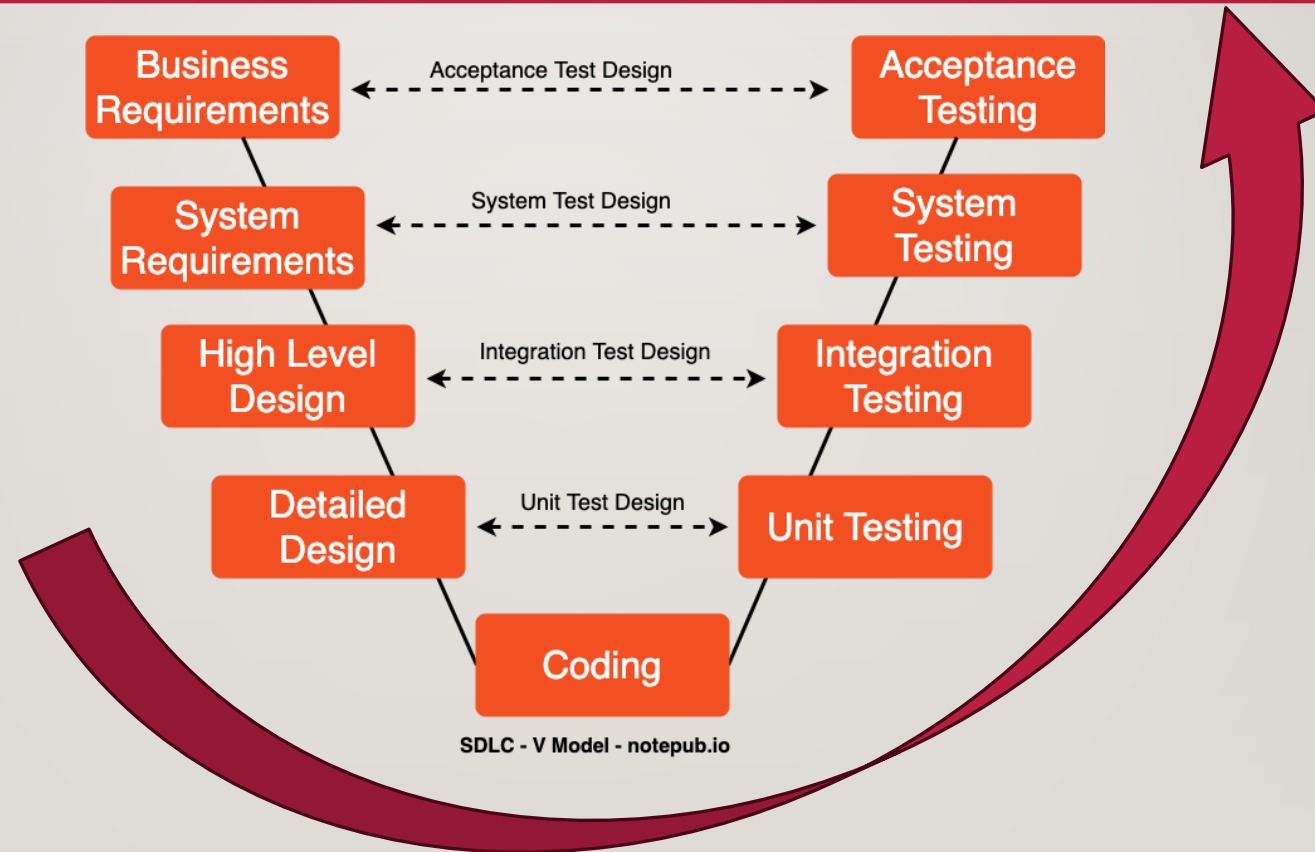
SYSTEM DESIGN PRESENTATION BY EACH TEAM



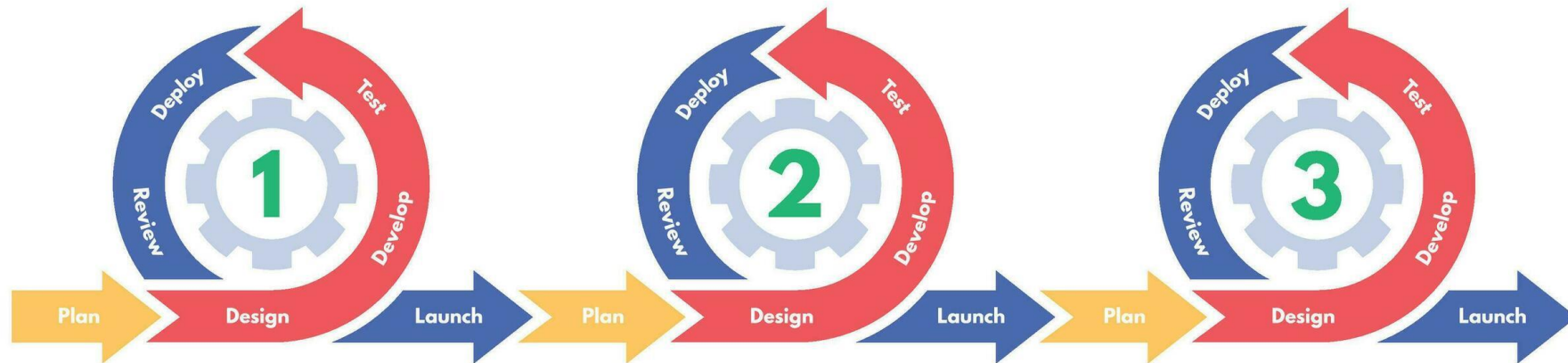
DETAIL DESIGN TO USER ACCEPTANCE



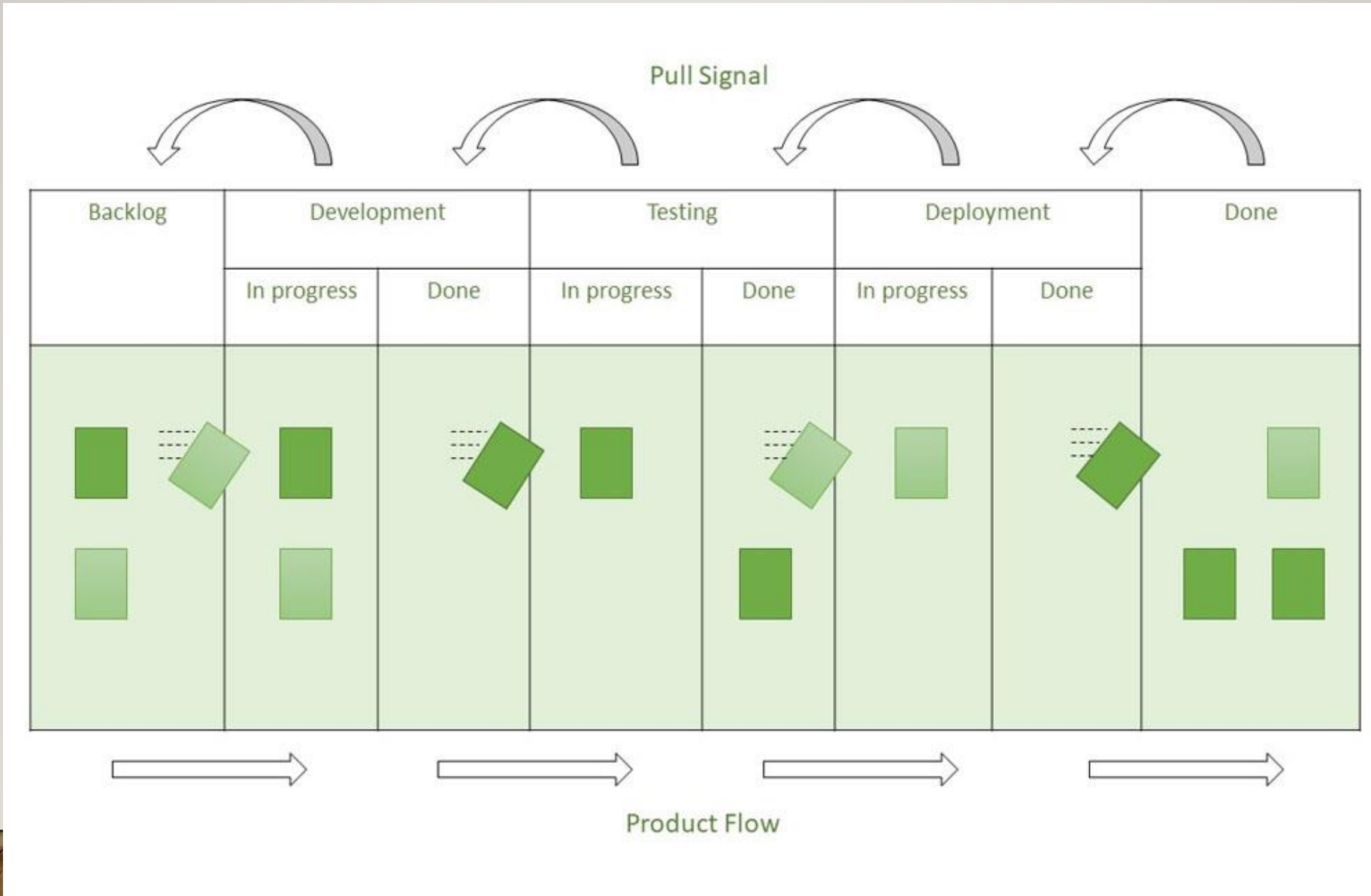
SW DEVELOPMENT PROCESS



AGILE DEVELOPMENT

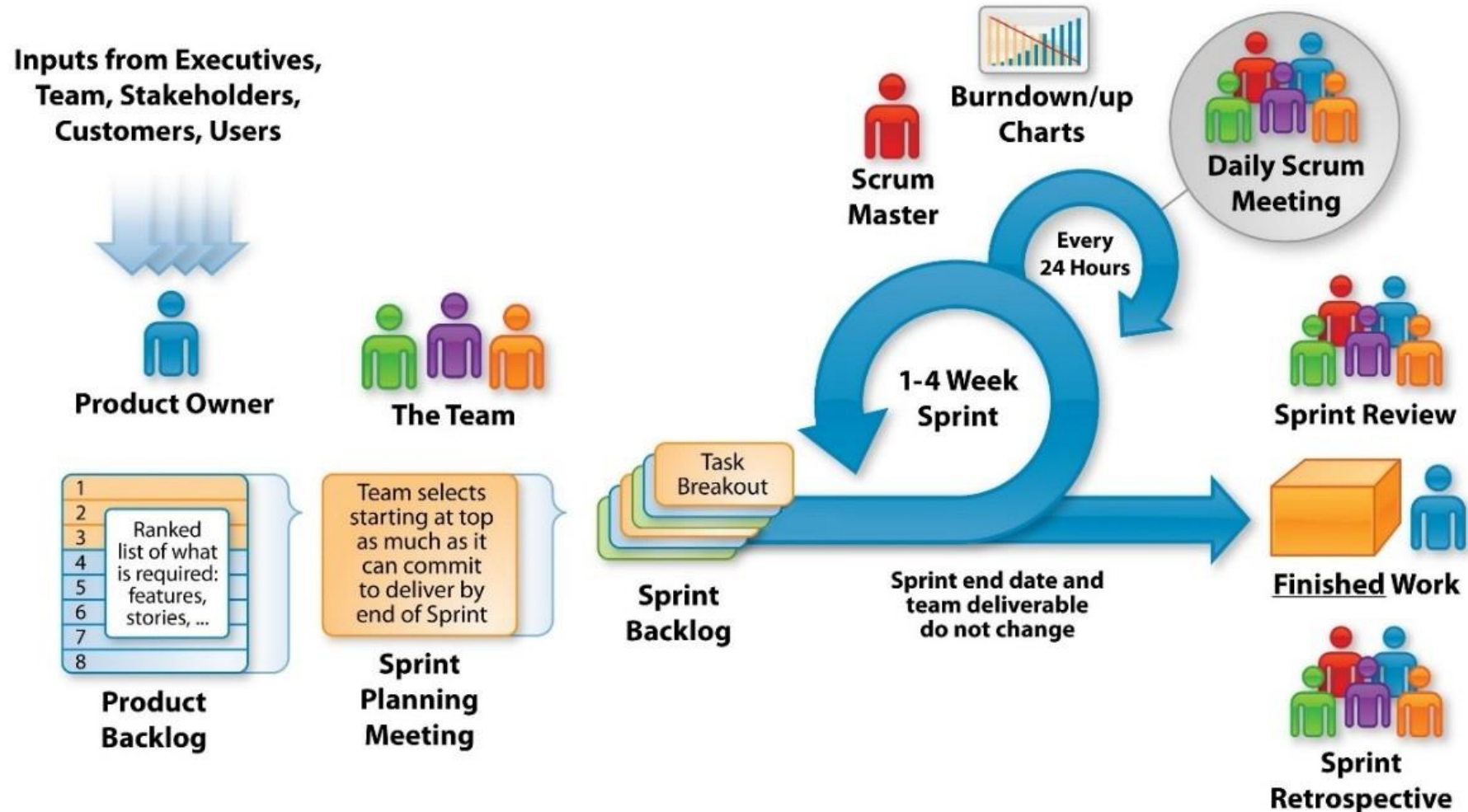


KANBAN METHODOLOGY





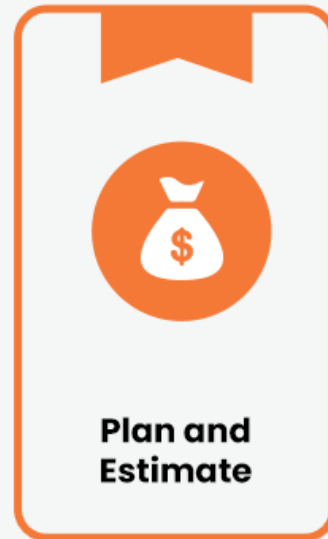
The Agile - Scrum Framework



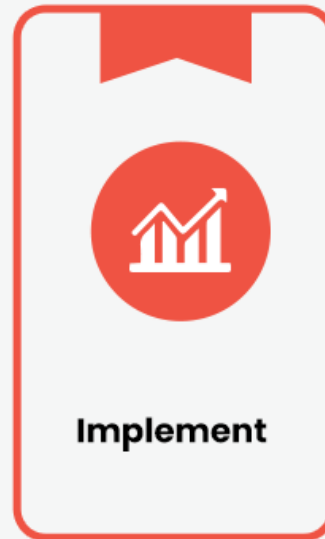
5 Stages of Scrum Sprint



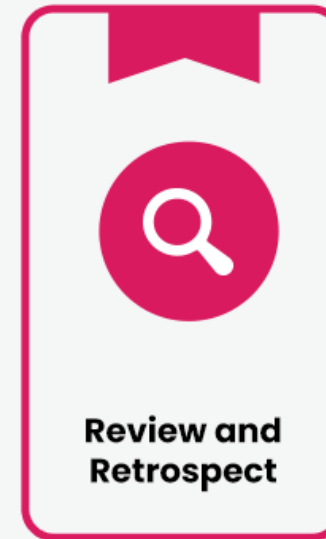
This phase includes the processes related to the commencement of a project, such as a scope and objectives, creating and distributing its charter, and taking other steps to guarantee success.



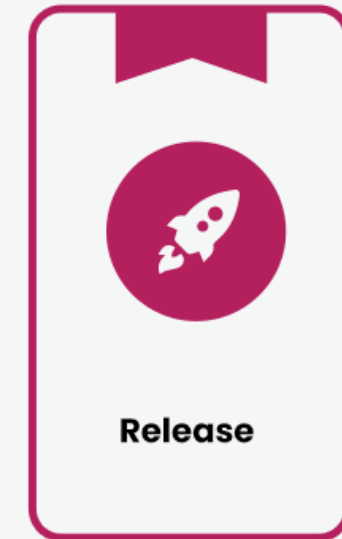
This phase involves planning and estimating processes, including creating user stories, approving, assessing, committing user stories, creating tasks, evaluating tasks, and creating a Sprint backlog.



This phase is about executing the tasks and activities to create a product. These activities include building the various outputs, conducting daily standup meetings, and grooming the product backlog.

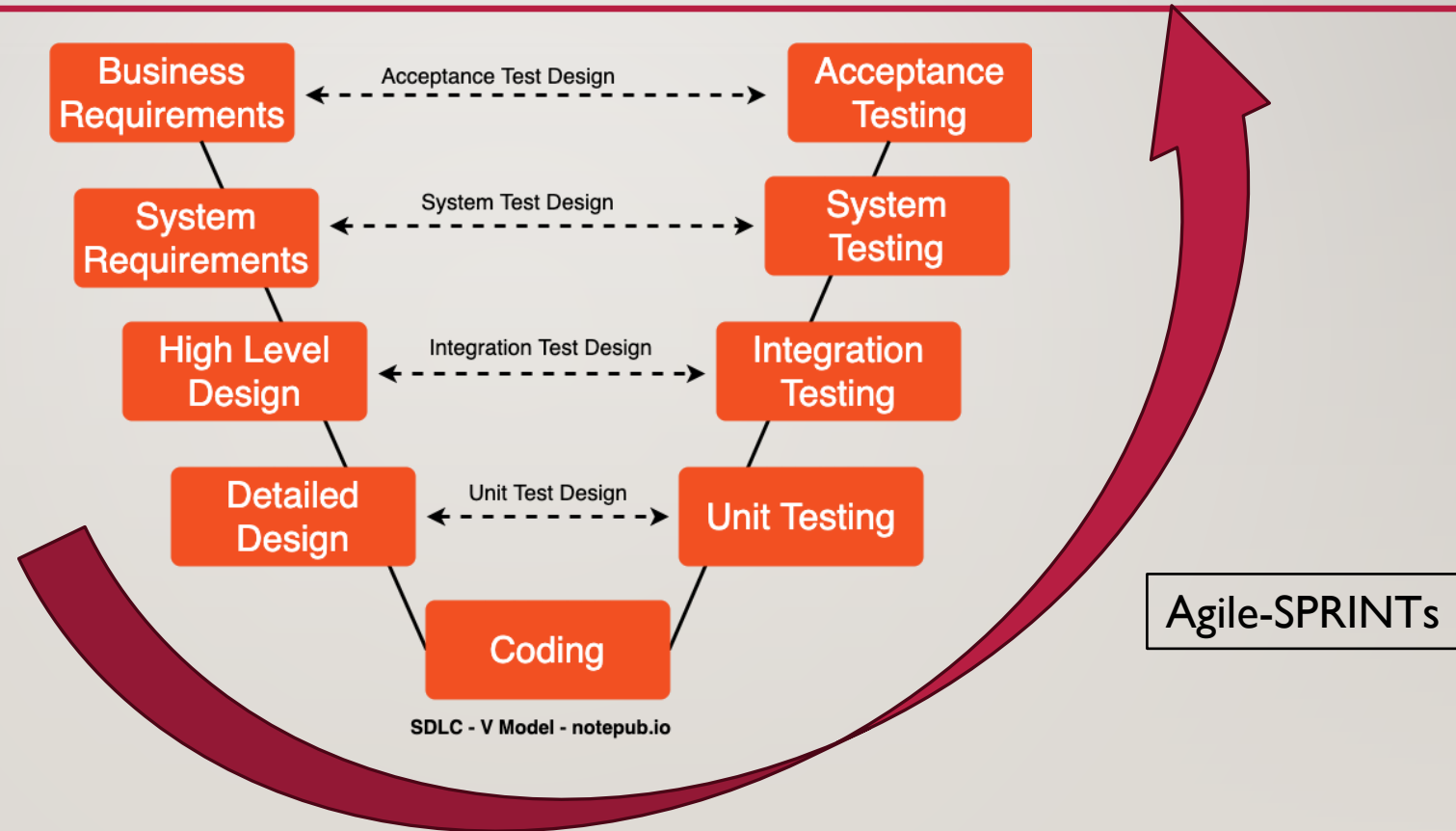


This stage of the project lifecycle is concerned with evaluating what has been accomplished so far, whether the team has worked to plan, and how it can do things better in the future.



This stage highlights delivering the accepted deliverables to the customer and determining, documenting, and absorbing the lessons learned during the project.

SW DEVELOPMENT PROCESS



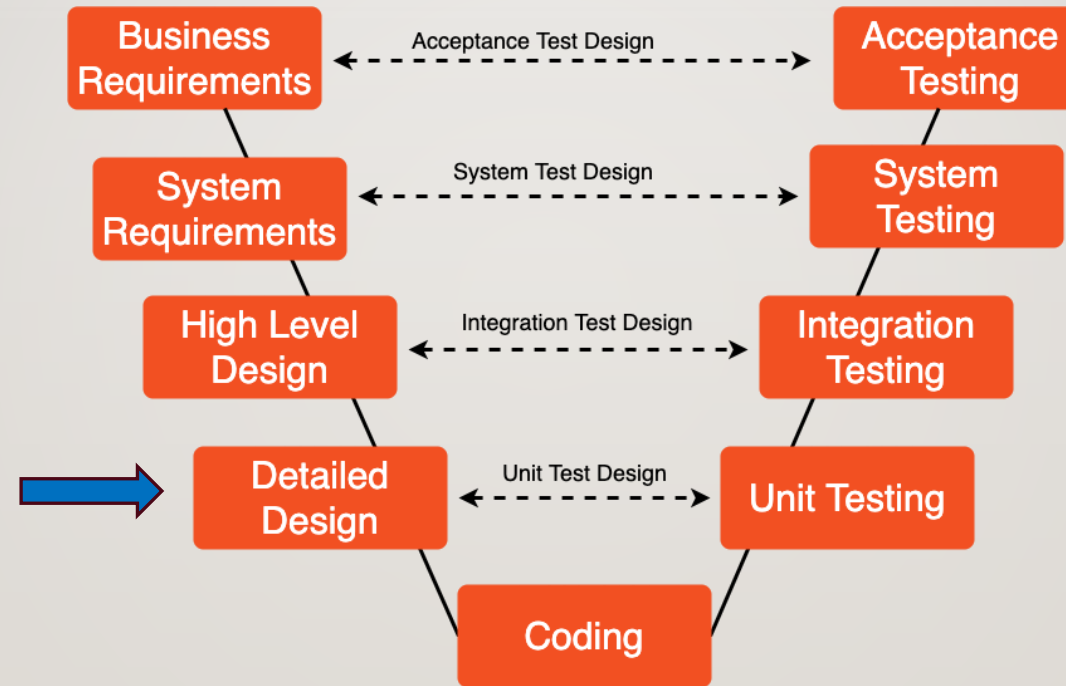
PROJECT SPRINTS

- Security Alert
 - Object Detection
- System Monitor
 - Display Security Camera and info
 - Display AMR Camera and info
 - Store, display, and report Alerts
- AMR Controller
 - Movement (SLAM)
 - Target Acquisition (Obj. Det.) and Tracking

SECURITY ALERT SPRINT



SPRINT I - SECURITY ALERT



SDLC - V Model - notepub.io

TEAM EXERCISE 5

Perform Detail Design of Security Alert Module using Process Flow Diagram

DETAIL DESIGN REVIEW BY EACH TEAM

Using the process flow diagram present team's design

EXAMPLE DETAILED DESIGN DOCUMENT

Detailed Design Document: AMR Navigation and Threat Detection

Project Title: Autonomous Mobile Robot (AMR) Security System

Version: 1.0

Date: [Insert Date]

1. Overview

This document outlines the detailed design for the Autonomous Mobile Robot (AMR) navigation and threat detection components. It covers the architecture, algorithms, data processing, and system interactions necessary to enable autonomous navigation within a secure area and real-time threat detection using onboard sensors.

2. System Architecture

The AMR system relies on onboard hardware (e.g., sensors, cameras, Jetson-Orin processor) and software (ROS2, OpenCV, YOLO) for autonomous navigation and real-time threat detection. All processing occurs locally on the AMR, with the capability to transmit alerts to a monitoring PC via Wi-Fi.

상세 설계 문서: AMR 네비게이션 및 위협 탐지

프로젝트 제목: 자율 이동 로봇(AMR) 보안 시스템

버전: 1.0

날짜: [날짜 삽입]

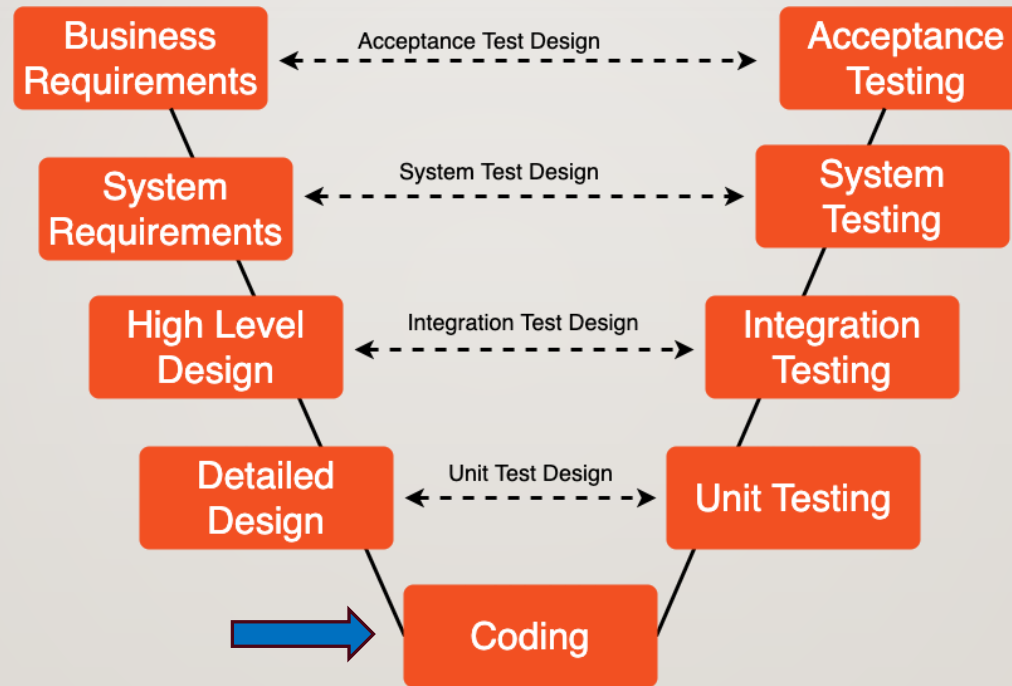
1. 개요

이 문서는 자율 이동 로봇(AMR)의 네비게이션 및 위협 탐지 구성 요소에 대한 상세 설계를 다룹니다. 자율 네비게이션과 실시간 위협 탐지를 위해 온보드 센서를 사용하는 데 필요한 아키텍처, 알고리즘, 데이터 처리 및 시스템 상호작용이 포함되어 있습니다.

2. 시스템 아키텍처

AMR 시스템은 자율 네비게이션 및 실시간 위협 탐지를 위해 온보드 하드웨어(예: 센서, 카메라, Jetson-Orin 프로세서)와 소프트웨어(ROS2, OpenCV, YOLO)를 활용합니다. 모든 처리는 AMR 내에서 로컬로 수행되며, 잠재적인 위협이 감지되면 Wi-Fi를 통해 모니터링 PC로 알림을 전송할 수 있습니다.

SPRINT I - SECURITY ALERT



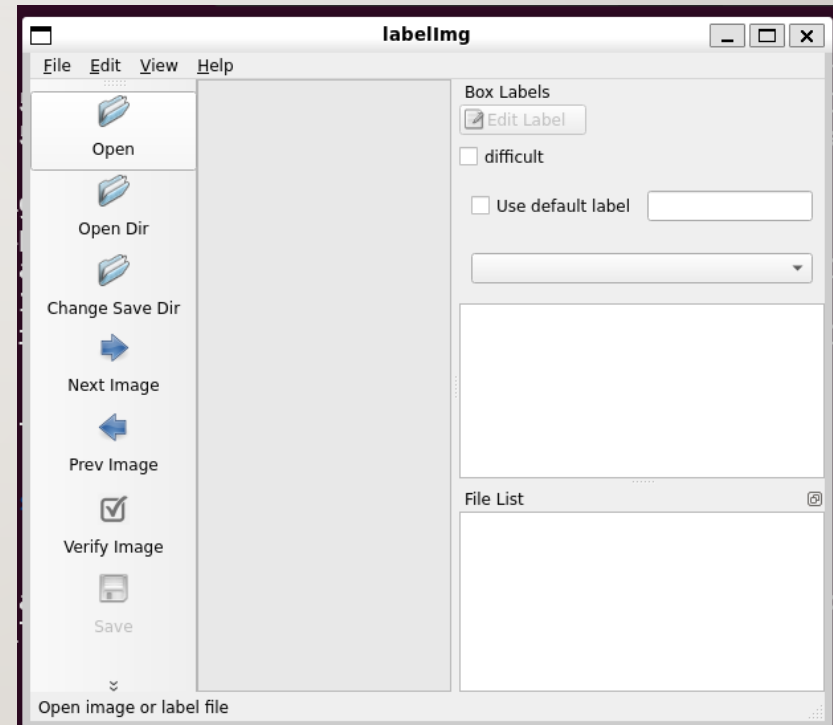
SDLC - V Model - notepub.io

CODING HINTS

- Image Capture
- Data Labelling : Preprocessing

CODING HINTS

- Data Labelling : use previously installed Labellmg
- Or



CODING HINTS

- Data Labelling : Labellmg

라벨링 순서

1. 이미지파일 불러오기 (Open Dir)
2. 저장형식 변경 (PascalVOC, YOLO)
3. 이미지 선택
4. 바운딩 박스 그리기(create rectbox)
5. Class 지정
6. 저장경로 생성 및 변경(Change Save Dir)
7. 저장(Save)

단축키

Ctrl + u	Load all of the images from a directory
Ctrl + r	Change the default annotation target dir
Ctrl + s	Save
Ctrl + d	Copy the current label and rect box
Ctrl + Shift + d	Delete the current image
Space	Flag the current image as verified
w	Create a rect box
d	Next image
a	Previous image
del	Delete the selected rect box
Ctrl++	Zoom in
Ctrl--	Zoom out
↑→↓←	Keyboard arrows to move selected rect box

CODING HINTS

- Image Capture
- Data Labelling
- Preprocessing

```
0.capture_image.py
1.cont_capture_image.py
2.labellmg.exe
3.create_data_dirs.py
4.move_image.py
5.move_labels.py
```

PERFORM DATA COLLECTION FOR SECURITY ALERT



CODING HINTS

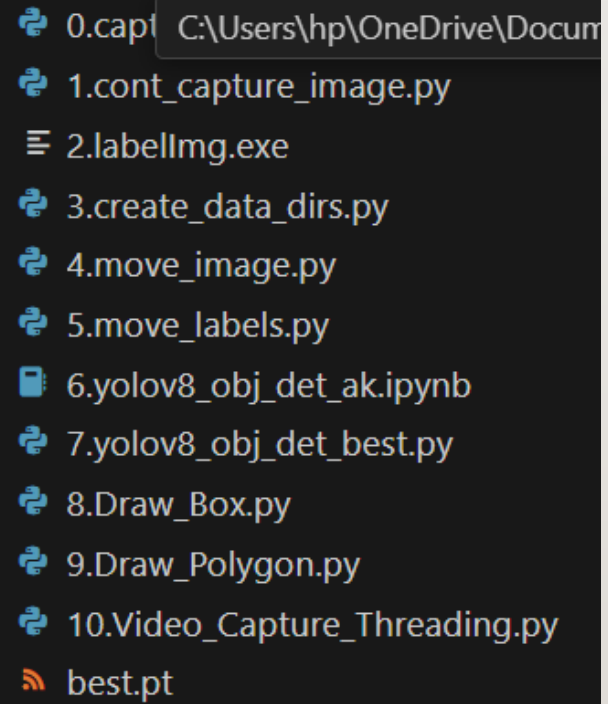
- Yolo8 Object Det.

```
0.capture_image.py
1.cont_capture_image.py
2.labellmg.exe
3.create_data_dirs.py
4.move_image.py
5.move_labels.py
6.yolov8_obj_det_ak.ipynb
7.yolov8_obj_det_best.py
```

HOW DID YOU DEFINE SECURE AREA?



CODING HINTS



A screenshot of a file explorer window with a dark background. The address bar at the top shows the path "C:\Users\hp\OneDrive\Docum". Below the address bar, a list of files and folders is displayed, each with a small icon to its left. The files are numbered 0 through 10, with the last one being "best.pt".

- 0.capt C:\Users\hp\OneDrive\Docum
- 1.cont_capture_image.py
- 2.labellmg.exe
- 3.create_data_dirs.py
- 4.move_image.py
- 5.move_labels.py
- 6.yolov8_obj_det_ak.ipynb
- 7.yolov8_obj_det_best.py
- 8.Draw_Box.py
- 9.Draw_Polygon.py
- 10.Video_Capture_Threading.py
- best.pt

PORTING TO ROS



ROS2 DEVELOPMENT WORKSPACE

```
$ mkdir -p ~/ros2_ws/src
```

```
$ cd ~/ros2_ws/src
```

ROS2 DEVELOPMENT WORKSPACE

CREATE ROS VIRTUAL ENVIRONMENT

```
$ cd ~/ros2_ws
```

```
$ python3 -m venv <NAME>
```

```
$ source <NAME>/bin/activate
```

```
$ rosdep install --from-paths src --ignore-src -r -y
```

```
$ colcon build
```

```
$ source install/setup.bash
```

add activation to .bashrc or setup.bash

- echo "source
~/ros2_ws/venv/bin/activate" >>
~/.bashrc

ROS2 DEVELOPMENT WORKSPACE

```
$ cd ~/ros2_ws/src
```

```
$ ros2 pkg create --build-type  
ament_python <my_package>
```

Or

```
$ ros2 pkg create <package_name> --  
build-type ament_python --dependencies  
rcldpy std_msgs ament_index_python
```

```
my_package/  
├─ package.xml          # Package metadata and dependencies  
├─ setup.py             # Build instructions for Python packages  
├─ setup.cfg            # Optional, configures metadata for setuptools  
├─ launch/              # Launch files for starting nodes (optional)  
├─ config/              # Configuration files (optional)  
├─ resource/            # Empty file matching package name for ament index  
├─ my_package/          # Python package directory (contains code)  
│   └─ __init__.py      # Makes this directory a Python package  
│   └─ my_node.py       # Example Python node  
└─ msg/                 # Message definitions (optional)
```

ROS2 DEVELOPMENT WORKSPACE

Write your code below the `my_package/` directory under `my_package/ package directory`

```
my_package/
├─ package.xml          # Package metadata and dependencies
├─ setup.py             # Build instructions for Python packages
├─ setup.cfg            # Optional, configures metadata for setuptools
├─ launch/              # Launch files for starting nodes (optional)
├─ config/              # Configuration files (optional)
├─ resource/            # Empty file matching package name forament inc
├─ my_package/          # Python package directory (contains code)
│   └─ __init__.py      # Makes this directory a Python package
│   └─ my_node.py       # Example Python node
└─ msg/                 # Message definitions (optional)
```


ROS2 DEVELOPMENT WORKSPACE

```
$ cd ~/ros2_ws
```

```
$ colcon build
```

or

```
$ colcon build --packages-select  
  <package_name>
```

or

```
$ colcon build --packages-select  
  <package_name> --packages-skip-up-to-  
  date
```

```
$ rm -rf build/<package_name>  
  install/<package_name> log
```

```
$ colcon build --packages-select  
  <package_name>
```

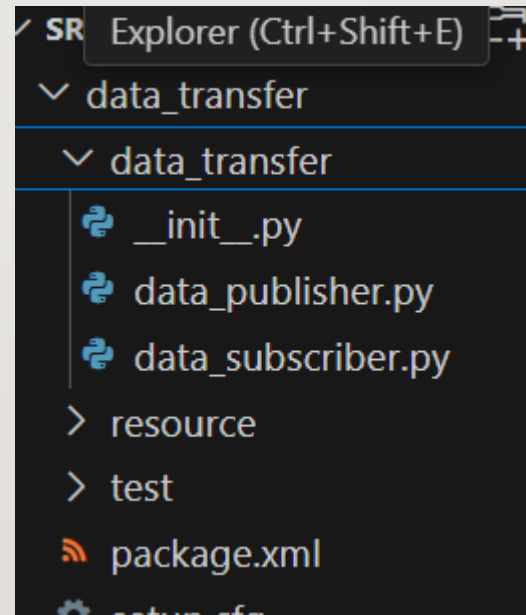
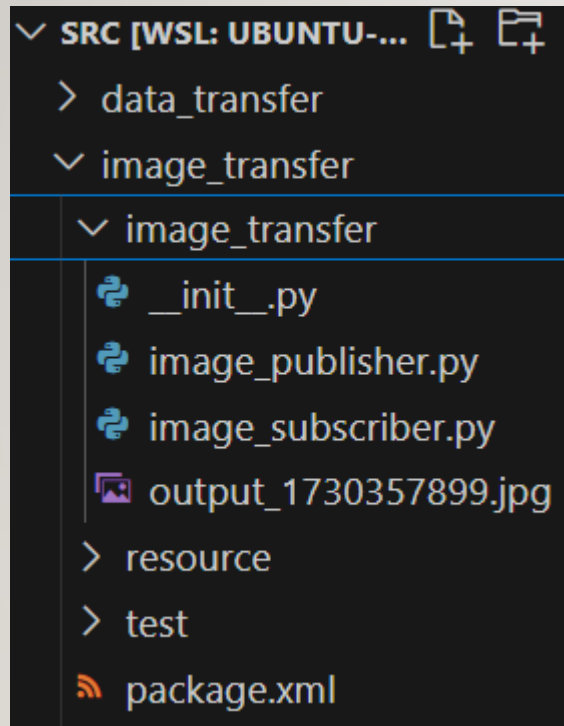
```
workspace/      # Root of the workspace  
├─ src/         # Source code (ROS packages)  
├─ build/       # Build files (generated by colcon)  
├─ install/     # Installed packages and setup scripts  
└─ log/         # Build logs
```

ROS2 DEVELOPMENT WORKSPACE

```
$ echo "source  
~/ros2_ws/install/setup.bash" >>  
~/.bashrc
```

```
$ source ~/.bashrc
```

ROS HINTS



\$ ros2 interface list

ROS HINTS

- Edit setup.py under <package_name>
directory add entry for each node

```
entry_points={ 'console_scripts':  
[ '<command_name> =  
<package_name>.<code_filename>:main', },
```

<command_name> is used when ros2 run
is executed i.e. data_publisher

```
entry_points={  
    'console_scripts': [  
        'data_pub = data_transfer.data_publisher:main',  
        'data_sub = data_transfer.data_subscriber:main',  
    ],  
}
```


ROS HINTS

\$ source ~/ros2_ws/install/setup.bash

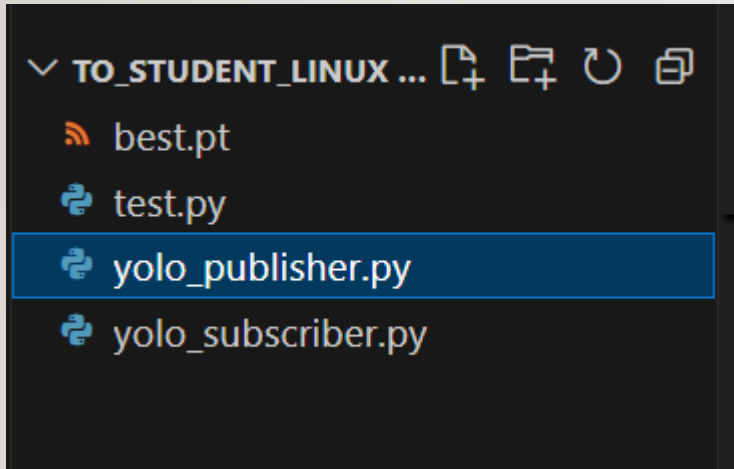
\$ sudo apt update

\$ ros2 run <package_name>
 <command_name>

\$ sudo apt install terminator

```
391 ros2 run data_transfer data_pub
```

ROS HINTS



\$ Ros2 run rqt_graph rqt_graph

\$ ros2 node list

\$ ros2 node info <node_name>

\$ ros2 topic list

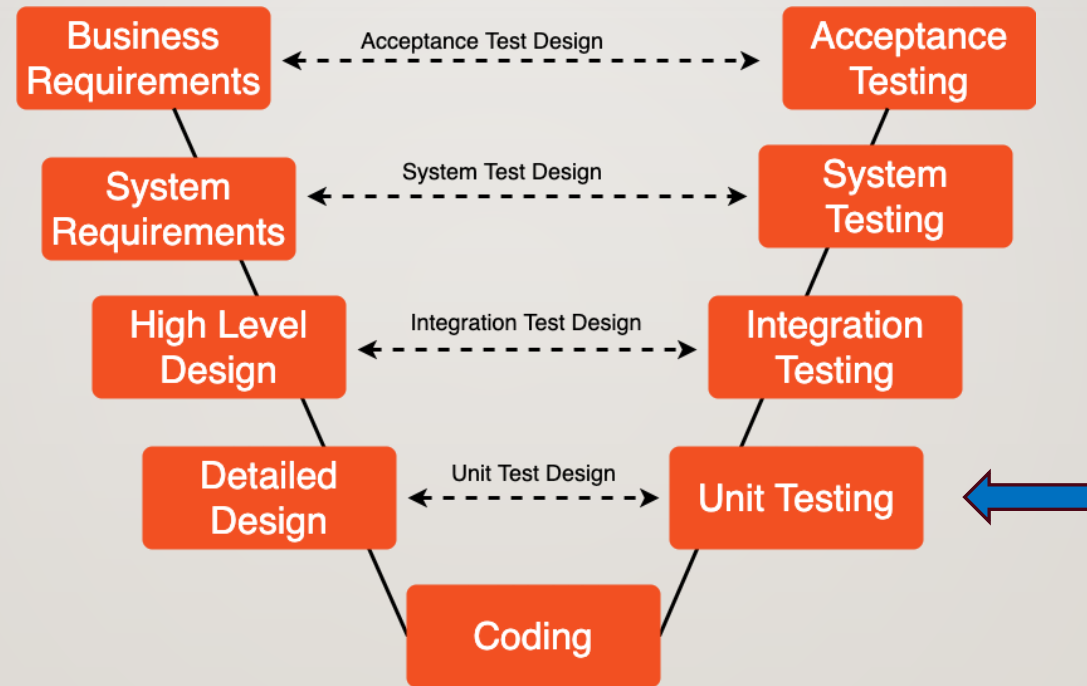
\$ ros2 topic info <topic_name>

\$ ros2 topic echo /chatter

\$ ros2 interface list

\$ ros2 interface show
<package_name>/msg/<MessageName>

SPRINT I - SECURITY ALERT



SDLC - V Model - notepub.io

EXPECTED OUTCOME

- Successful object detection
- ROS Nodes, and Topics created to send and display images and data

TEAM EXERCISE 6

Perform coding and testing of Security Alert Module

RESULTS & CODE REVIEW BY EACH TEAM

Show actual results against the expected results and explain the code written