

GOOD MORNING!

早上好!

안녕하세요!

DAY 6



DAY I

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

DAY 2/3 (MINI PROJECT)

- Yolo객체 인식 모델 활용과 성능 평가 방법 이해
- Custom Dataset과 Fine Tuning으로 자체 객체 인식 모델 구현 및 평가
- (Optional)경량화 모델 등 개별 요구사항에 적합한 모델 탐색 및 성능 검증
- YOLOv8 기반 데이터 수집/학습/deploy (Detection Alert)
 - 감시용 데이터 수집(bus, truck, tank 등)
 - 감시용 데이터 라벨링
 - YOLOv8 기반 학습
 - YOLOv8 Object Detection
- Porting to ROS
 - Create Detection Alert Node
 - Generate Topics to send image and Obj. Det. results
 - Create Subscriber node and display image and print data from the Topic

DAY 3/4 (MINI PROJECT)

- AMR(Autonomous Mobile Robot)
Turtlebot4 개발 환경 구축
- 로봇 개발 환경에 완성 모델 서빙 및 테스트 / 로봇 H/W, 제반 환경의 한계점 도출
 - Tracking 데이터 수집(bus, truck, tank 등)
 - Tracking 데이터 라벨링
 - YOLOv8 기반 학습
 - YOLOv8 Object **Tracking**
- Turtlebot4 시뮬레이션 환경 구축
 - SLAM과 Map 생성 및 파라미터 튜닝 (Localization, AMCL)
 - AutoSLAM으로 맵 생성

DAY 5/6 (MINI PROJECT)

- Turtlebot4 API를 활용한 Initial Pose Navigate_to Pose 구현
- Turtlebot4 API를 활용한 Navigate_Through_pose, Follow Waypoints 구현
- 로봇 개발 환경에 적용 및 테스트 / 로봇 H/W, 제반 환경의 한계점 도출
- AMR기반 카메라 인식 autonomous driving 시스템 with obstacle avoidance 구축 (AMR Controller)
 - Digital Mapping of environment
 - Goal Setting and Obstacle Avoidance using Navigation
 - Object Tracking w/ AMR camera
 - Control logic between navigation/obj. tracking/ obj. following (teleop)
- Porting to ROS
 - Create AMR Controller Node
 - Create and send Obj.Tracking Image and data to Sysmon
- Integrate and test with Detection

DAY 6 (MINI PROJECT)

- 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
- 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

DAY 6 (MINI PROJECT)

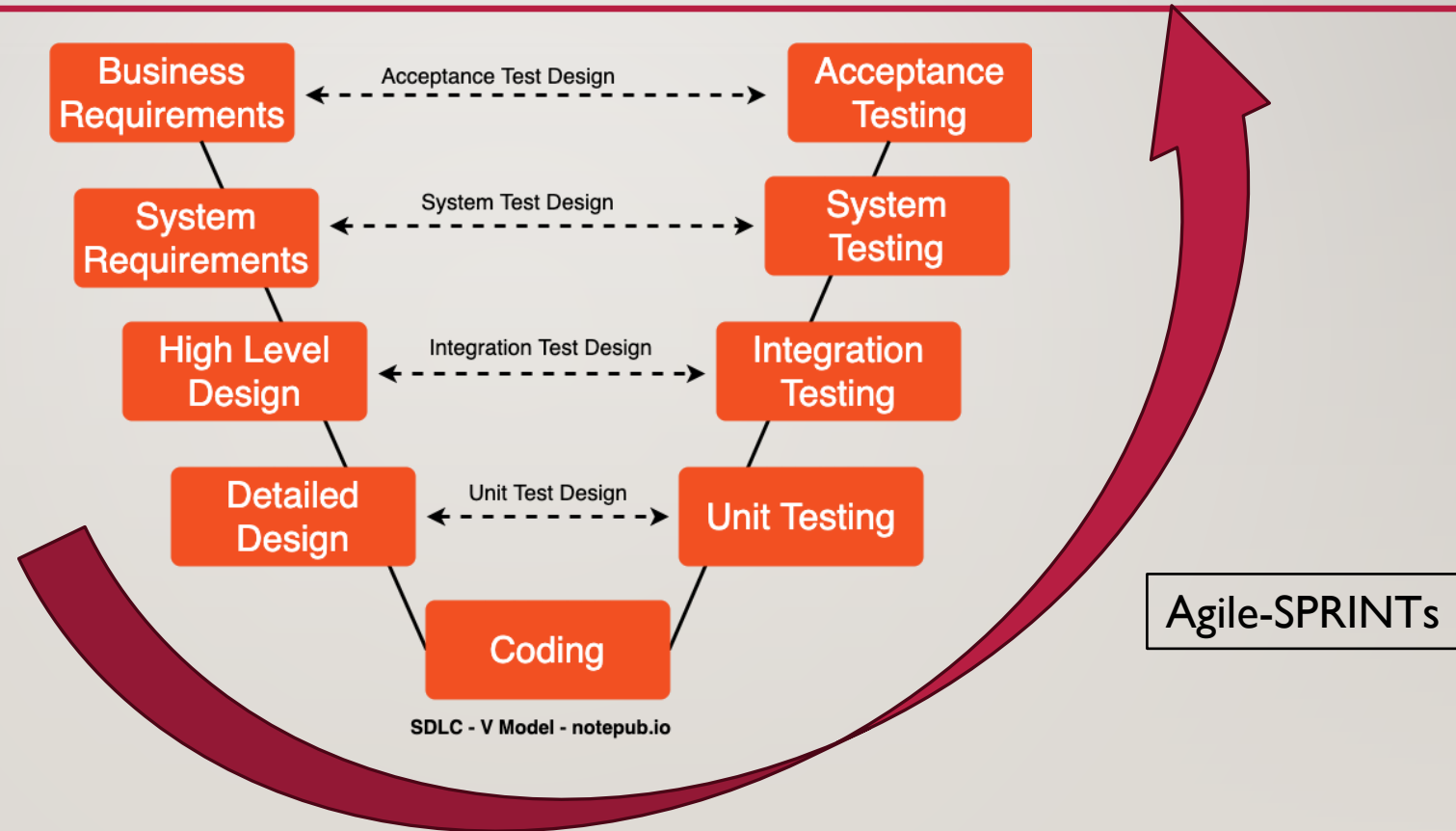
- Porting to ROS
 - Update Sysmon Node code
 - Update the database with received Obj. Det. Data from Detection Alert Node
 - Display the content of DB on System Monitor web page
- And finally, Integration and Test of Detection Alert & System Monitor

프로젝트 RULE NUMBER ONE!!!

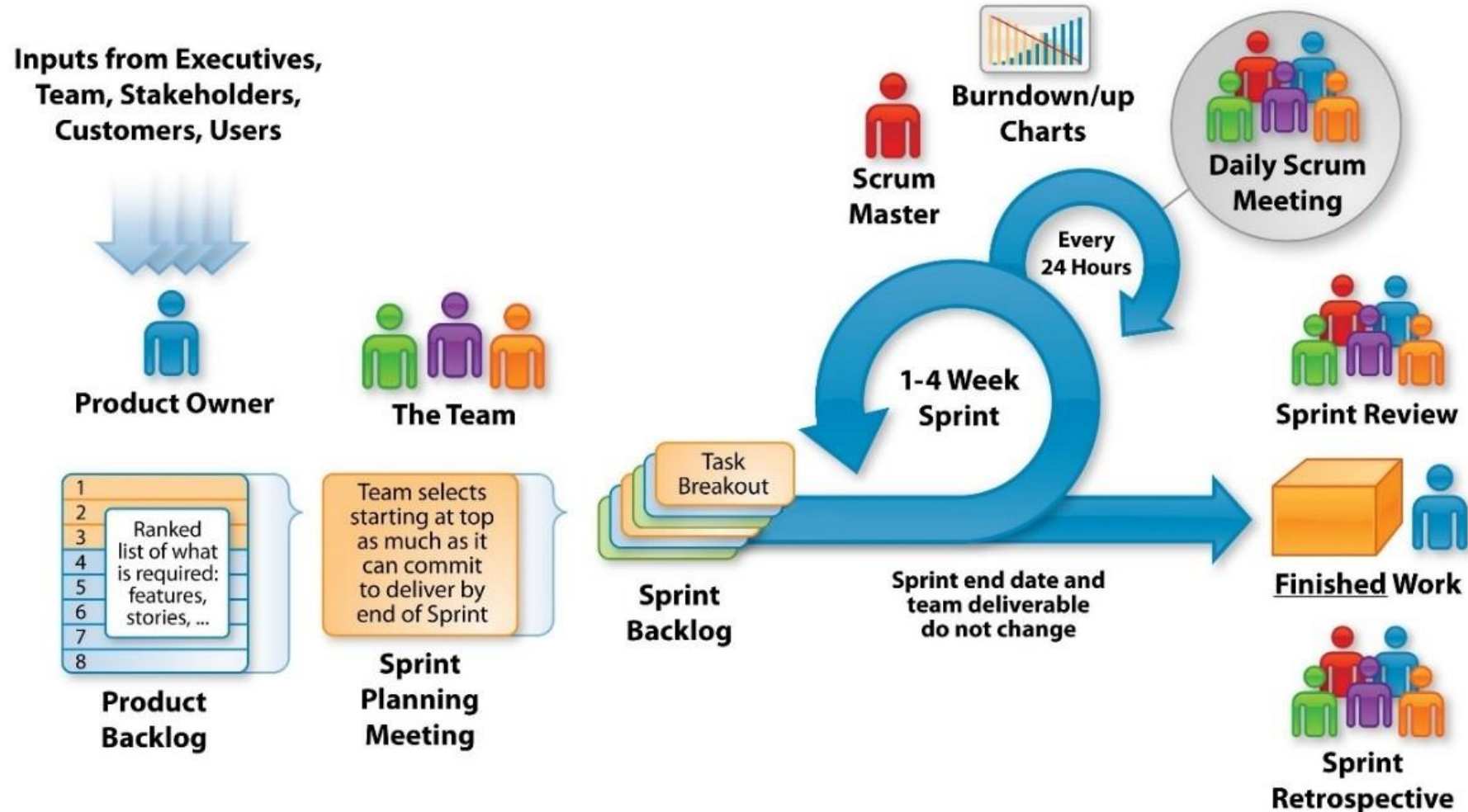
Have Fun Fun Fun!



SW DEVELOPMENT PROCESS



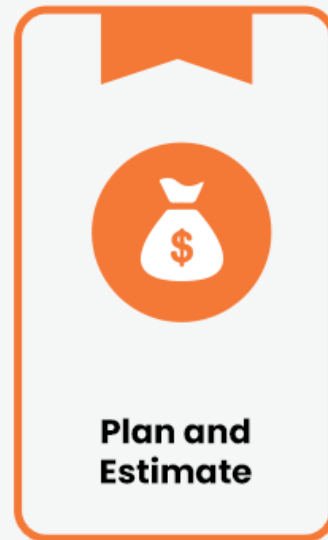
The Agile - Scrum Framework



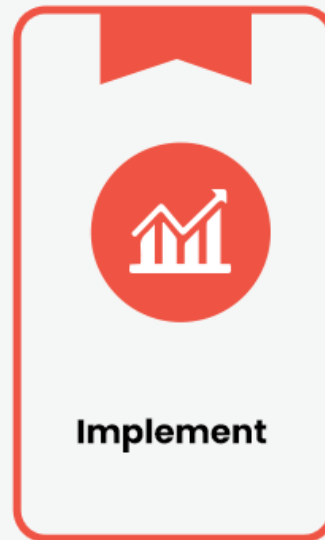
5 Stages of Scrum Sprint



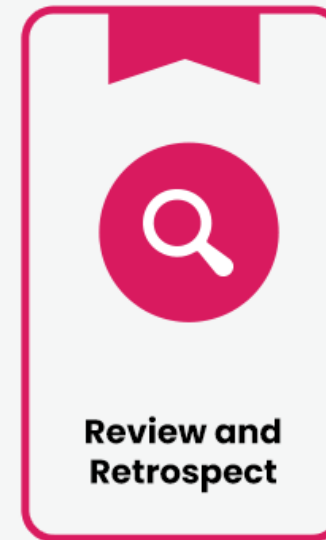
This phase includes the processes related to the commencement of a project, such as a scope and objectives, creating and distributing its charter, and taking other steps to guarantee success.



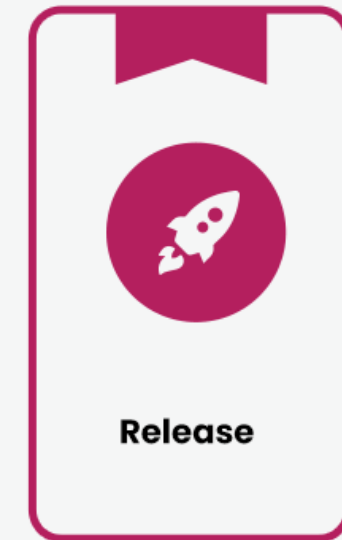
This phase involves planning and estimating processes, including creating user stories, approving, assessing, committing user stories, creating tasks, evaluating tasks, and creating a Sprint backlog.



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.

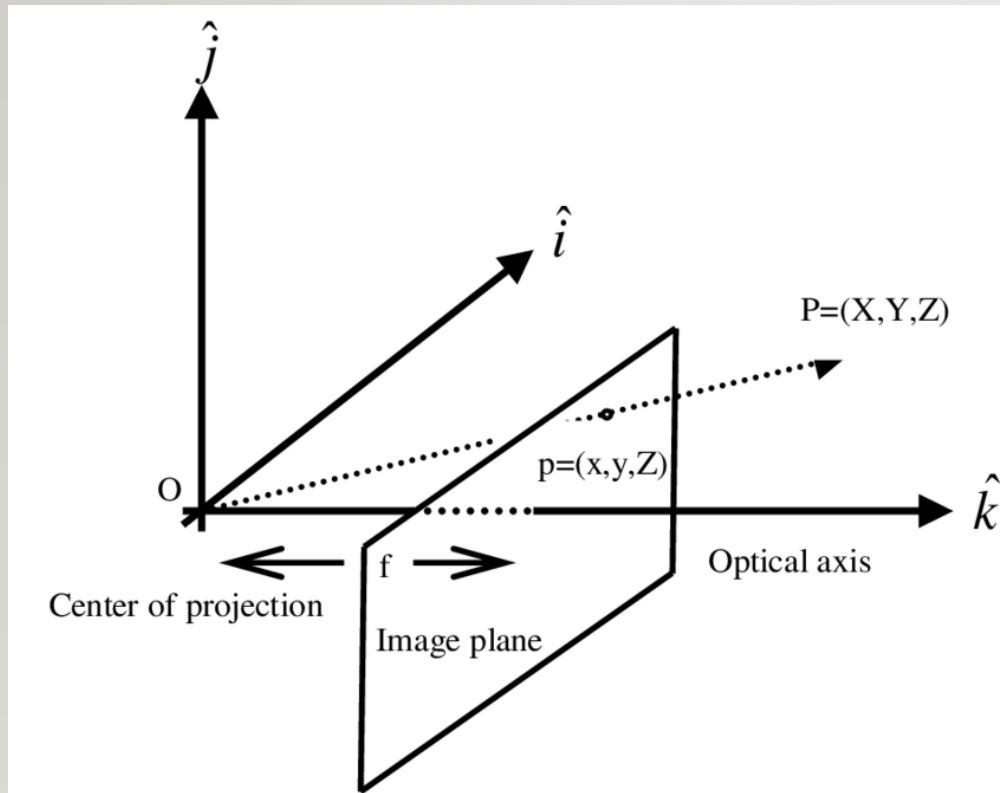


This stage highlights delivering the accepted deliverables to the customer and determining, documenting, and absorbing the lessons learned during the project.

USING DEPTH



CAMERA INTRINSIC AND REPROJECTION



$$X = \frac{(u - c_x) \cdot Z}{f_x}, \quad Y = \frac{(v - c_y) \cdot Z}{f_y}, \quad Z = Z$$

$F(x), F(y)$ = focal point

“How zoomed in are your images”

ROS TRANSFORM EXPLAINED

- [Transform Explained](https://indecisive-freedom-6e8.notion.site/Transform-Explained-1f38e215779c803ba95df4921332d670)

<https://indecisive-freedom-6e8.notion.site/Transform-Explained-1f38e215779c803ba95df4921332d670>



USING DEPTH (ROBOT)

```
day4
├── day4
│   ├── __init__.py
│   ├── 4_4_a_nav_to_pose.py
│   ├── 4_4_b_nav_through_poses.py
│   ├── 4_4_c_follow_waypoints.py
│   ├── 4_4_d_create_path.py
│   ├── 4_4_e_mail_delivery.py
│   ├── 4_4_f_patrol_loop.py
│   ├── 4_4_g_init_pose.py
│   ├── 4_4_h_send_goal_stop.py
│   ├── 4_4_i_send_waypoint.py
│   ├── 4_5_a_depth_checker.py
│   ├── 4_5_b_depth_to_3d.py
│   ├── 4_5_c_depth_to_nav_goal.py
│   ├── 4_5_d_nav_to_car.py
│   ├── depth.sh
│   ├── my_best.pt
│   └── oakd_pro_new.yaml
```

```
4_5_a_depth_checker.py
4_5_b_depth_to_3d.py
4_5_c_depth_to_nav_goal.py
4_5_d_nav_to_car.py
```

Exercise: using the simulation code develop the code for the actual robot

USING DEPTH (ROBOT)

- Terminal 1

```
$ ros2 launch turtlebot4_navigation  
localization.launch.py namespace:=robot<n>  
map:=$HOME/Documents/room/room_map.  
yaml
```

- Terminal 2

```
$ ros2 launch turtlebot4_navigation  
nav2.launch.py namespace:=/robot<n>
```

- Terminal 3

```
$ ros2 launch turtlebot4_viz  
view_robot.launch.py namespace:=/robot  
<n>
```


USING DEPTH (ROBOT)

```
4_5_a_depth_checker.py
4_5_b_depth_to_3d.py
4_5_c_depth_to_nav_goal.py
4_5_d_nav_to_car.py
```

Terminal 4

```
$ ros2 run day4 depth_to_3d --ros-args -r
__ns:=/robot<n> -r /tf:=/robot<n>/tf -r
/tf_static:=/robot<n>/tf_static

$ ros2 run day4 depth_to_goal --ros-args -
r __ns:=/robot<n> -r /tf:=/robot<n>/tf -
r /tf_static:=/robot<n>/tf_static
```

BEGINS SPRINTS
DETAIL DESIGN/CODING/TESTING



PROJECT SPRINTS

- Detection Alert

- Camera Capture
- Object Detection
- Send messages to other subsystems

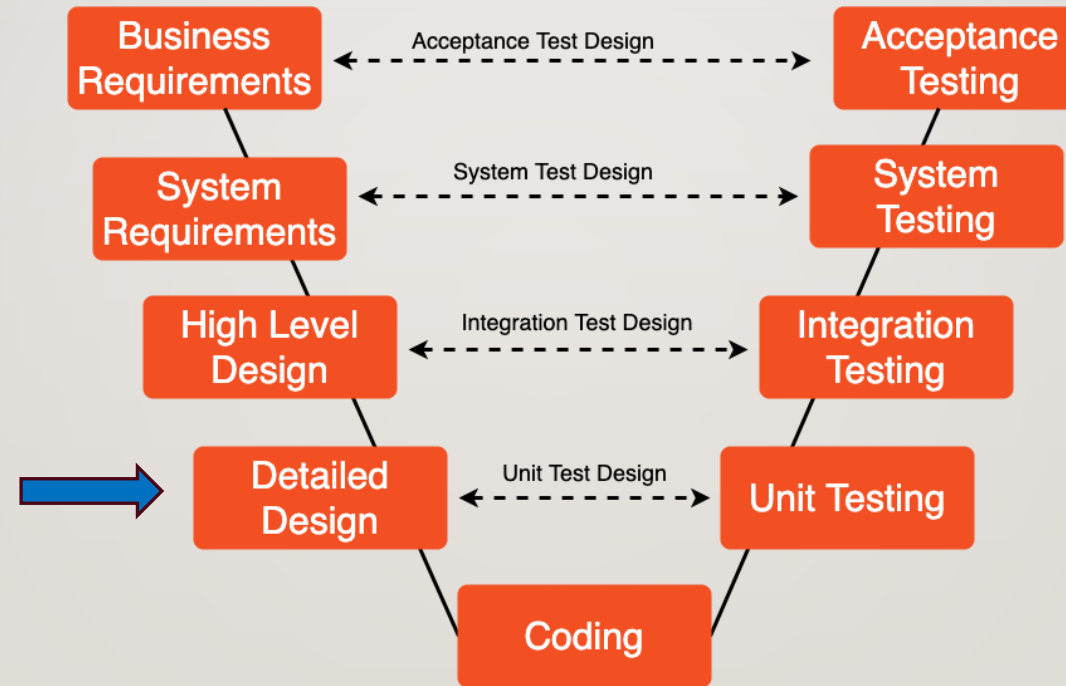
- AMR Controller

- Receive messages and act accordingly
- Move using (SLAM) with Obstruction avoidance
- Target Acquisition (Obj. Det.) and Tracking
- Follow target using camera and motor control

- System Monitor

- Receive and Display Detection Camera and info
- Receive and Display AMR Camera and info
- Store, display, and report Information and Alerts

SPRINT I - DETECTION ALERT



SDLC - V Model - notepub.io

TEAM EXERCISE 4

Perform Detail Design of Detection 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]

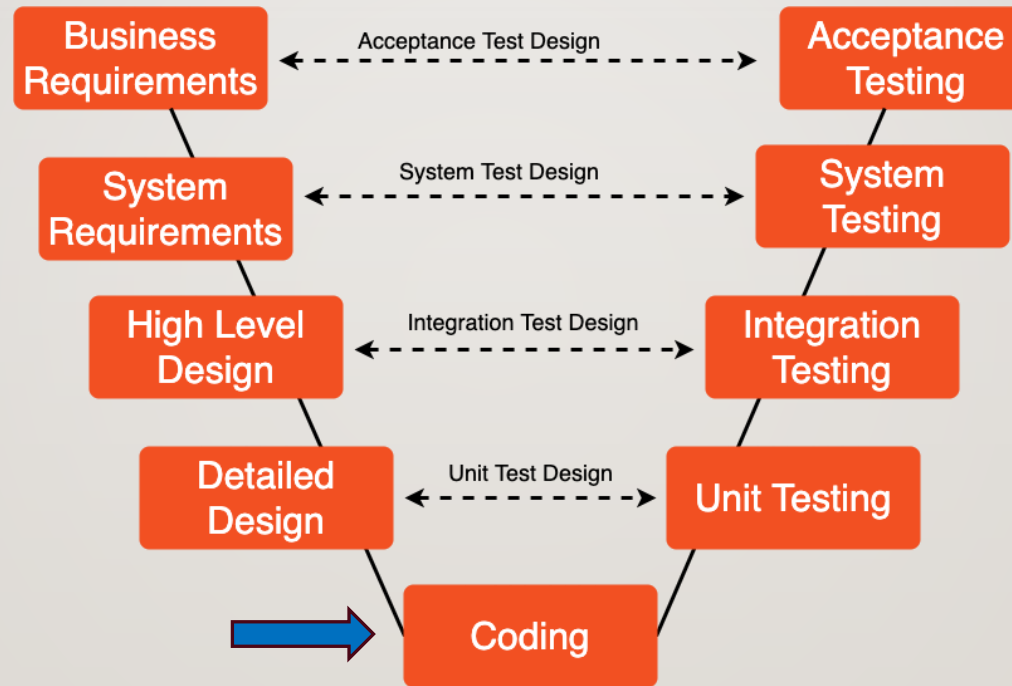
1. 개요

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

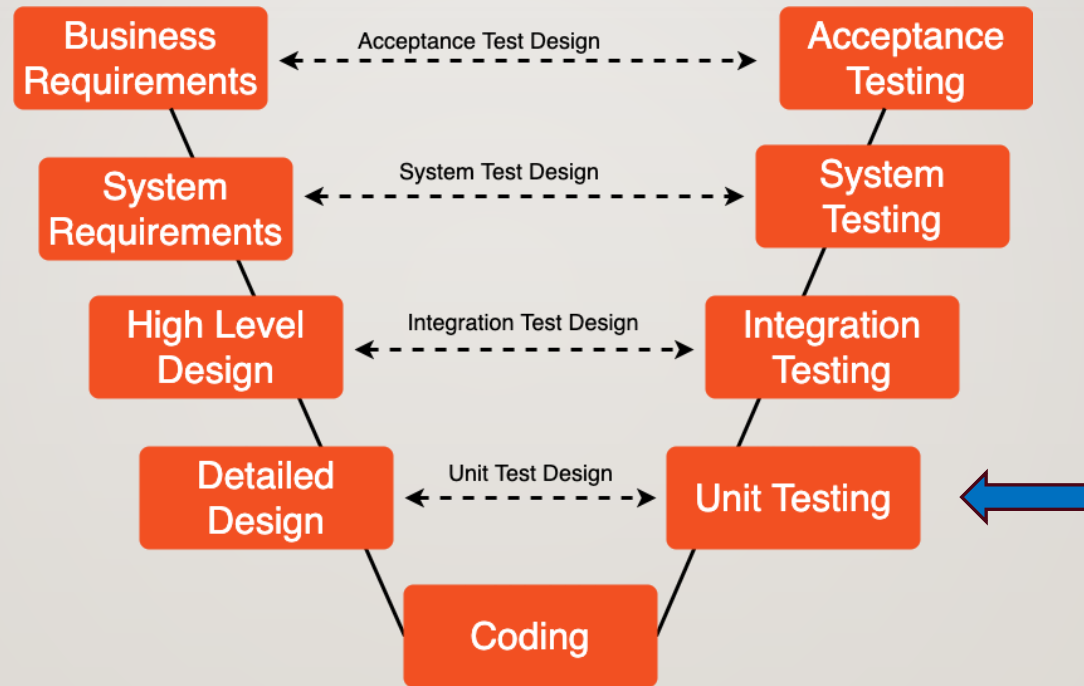
AMR 시스템은 자율 네비게이션 및 실시간 위협 탐지를 위해 온보드 하드웨어(예: 센서, 카메라, Jetson-Orin 프로세서)와 소프트웨어(ROS2, OpenCV, YOLO)를 활용합니다. 모든 처리는 AMR 내에서 로컬로 수행되며, 잠재적인 위협이 감지되면 Wi-Fi를 통해 모니터링 PC로 알림을 전송할 수 있습니다.

SPRINT I - DETECTION ALERT



SDLC - V Model - notepub.io

SPRINT I - DETECTION ALERT



SDLC - V Model - notepub.io

TEAM EXERCISE 5

Perform coding and testing of Detection Alert Module

EXPECTED OUTCOME

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

RESULTS & CODE REVIEW BY EACH TEAM

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

PROJECT SPRINTS

- Detection Alert

- Camera Capture
- Object Detection
- Send messages to other subsystems

- AMR Controller

- Receive messages and act accordingly
- Move using (SLAM) with Obstruction avoidance
- Target Acquisition (Obj. Det.) and Tracking
- Follow target using camera and motor control

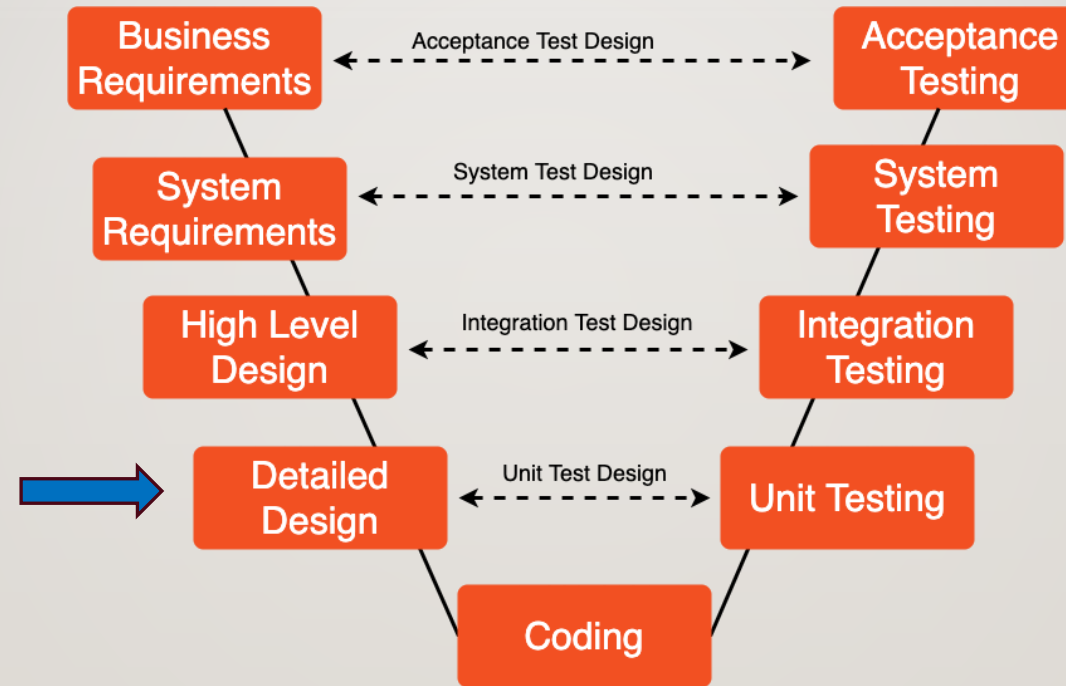
- System Monitor

- Receive and Display Detection Camera and info
- Receive and Display AMR Camera and info
- Store, display, and report Information and Alerts

AMR CONTROLLER SPRINT



SPRINT 2 – AMR CONTROLLER



SDLC - V Model - notepub.io

TEAM EXERCISE 6

Perform Detail Design of AMR Controller Module using Process Flow Diagram

DESIGN QUESTIONS:

- How do you start the robot?
 - Initial Position
- How do you find AMR current position and orientation?
- How do you sending Goals?
 - Single goal
 - Multiple goals
- How do you manual control of AMR Odometry?
 - How to move forward, backward, left and right???

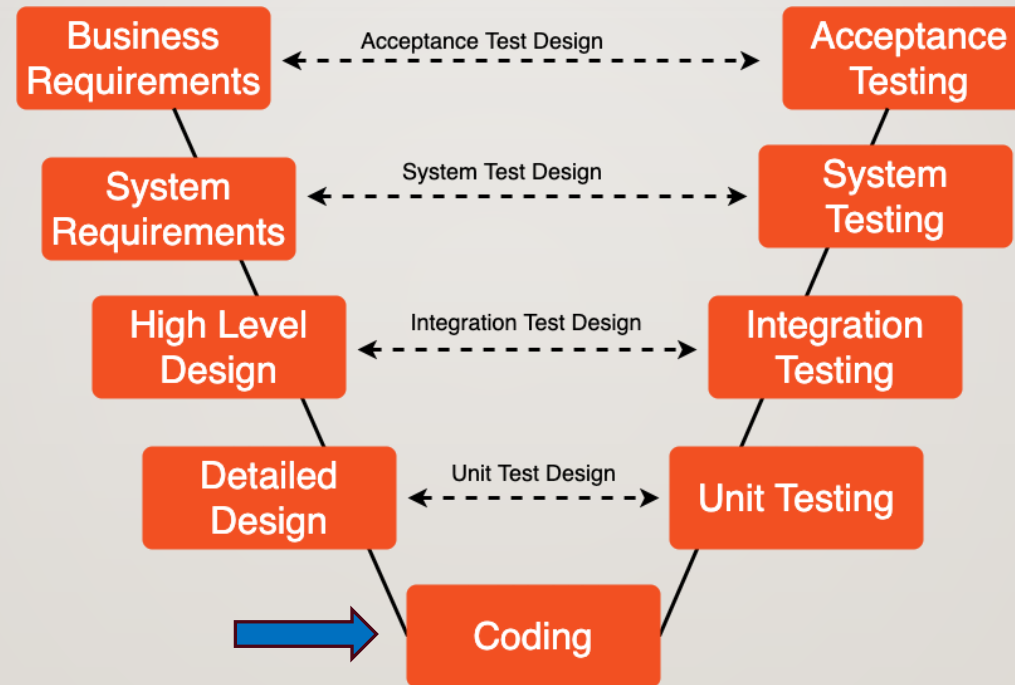
WHAT IS THE FOLLOW ALGORITHM?

- Left/Right?
- Forward/Backward?
- Velocity?
- Camera position?
- Depth? Local/Global Coordinate transform?

DETAIL DESIGN REVIEW BY EACH TEAM

Using the process flow diagram present team's design

SPRINT 2 – AMR CONTROLLER



SDLC - V Model - notepub.io

CODING HINTS:

- How do you start the robot?

```
# Start on dock
if not navigator.getDockedStatus():
    navigator.info('Docking before initialising pose')
    navigator.dock()

# Set initial pose
initial_pose = navigator.getPoseStamped([0.0, 0.0], TurtleBot4Directions.NORTH)
navigator.setInitialPose(initial_pose)
```

CODING HINTS:

- How do you find AMR current position and orientation?
 - Echo topic: `amcl_pose`
 - Use Rviz Publish Points & Echo `clicked_points`
 - Etc...

CODING HINTS:

- How do you sending Goals?
 - Single goal

```
# Wait for Nav2
navigator.waitForNav2Active()
```

```
# Set goal poses
# goal_pose = navigator.getPoseStamped([-13.0, 9.0], TurtleBot4Directions.EAST)
goal_pose = navigator.getPoseStamped([-1.7, -0.1], TurtleBot4Directions.EAST)

# x: 2.0924954414367676
# y: 4.481560230255127
# [x,y]=[-1.707,-0.106]

# Undock
navigator.undock()

# Go to each goal pose
navigator.startToPose(goal_pose)
```


CODING HINTS:

- How do you sending Goals?
 - Single goal
 - Multiple goals

```
# Set goal poses
goal_pose = []
goal_pose.append(navigator.getPoseStamped([-1.7, -0.1], TurtleBot4Directions.EAST))
goal_pose.append(navigator.getPoseStamped([-1.1, 1.6], TurtleBot4Directions.NORTH))
goal_pose.append(navigator.getPoseStamped([-1.0, 0.05], TurtleBot4Directions.NORTH_WEST))
```

```
# Undock
navigator.undock()

# Navigate through poses
navigator.startThroughPoses(goal_pose)
```

```
# Follow Waypoints
navigator.startFollowWaypoints(goal_pose)
```

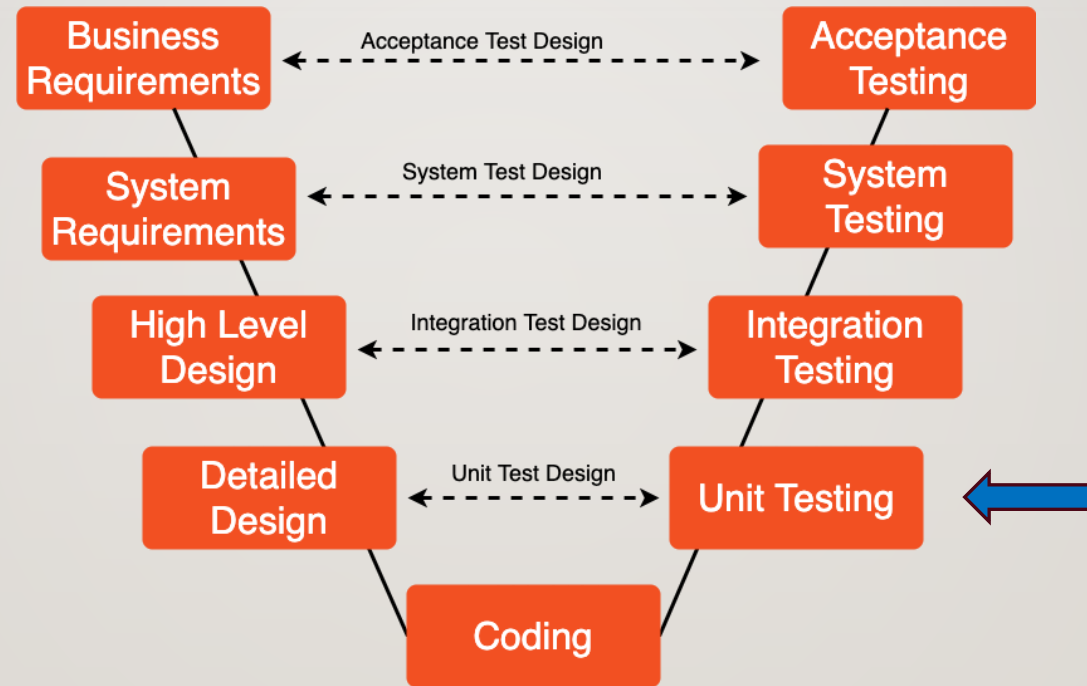
CODING HINTS:

- How do you manual control of AMR Odometry?
 - How to move forward, backward, left and right???
- Use teleops to move the robots
 - Echo topic: cmd_vel
- geometry_msgs.msg
 - Twist
 - twist.linear.x = <+/-n>
 - twist.angular.z = <+/-n>
 - self.cmd_publisher.publish(twist)

EXPECTED OUTCOME

AMR navigates to avoid obstacles, ignores dummies, track, and follow target

SPRINT 2 – AMR CONTROLLER



SDLC - V Model - notepub.io

TEAM EXERCISE 7

Perform coding and testing of AMR Controller Module

RESULTS & CODE REVIEW BY EACH TEAM

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

EXPECTED OUTCOME

- Detection Alert and AMR Controller able to pass topics for necessary actions between

TEAM EXERCISE 8

Perform integrate and test of Detection Alert and AMR Controller Modules

RESULTS & CODE REVIEW BY EACH TEAM

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

PROJECT SPRINTS

- Detection Alert

- Camera Capture
- Object Detection
- Send messages to other subsystems

- AMR Controller

- Receive messages and act accordingly
- Move using (SLAM) with Obstruction avoidance
- Target Acquisition (Obj. Det.) and Tracking
- Follow target using camera and motor control

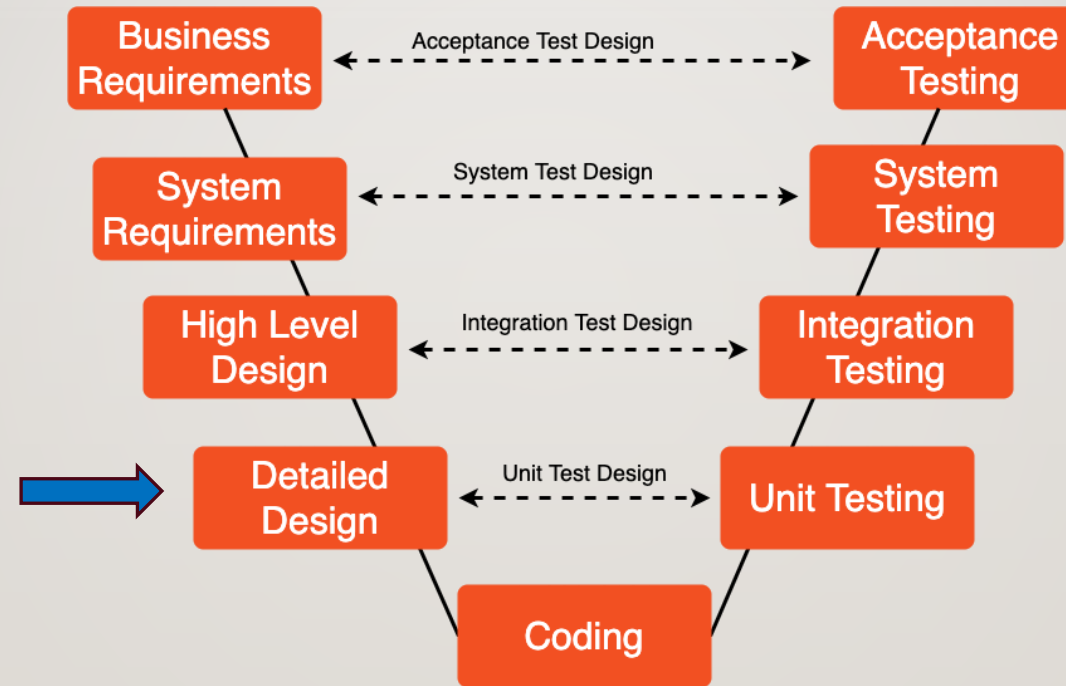
- System Monitor

- Receive and Display Detection Camera and info
- Receive and Display AMR Camera and info
- Store, display, and report Information and Alerts

SYSTEM MONITOR SPRINT



SPRINT 3 – SYSTEM MONITOR



SDLC - V Model - notepub.io

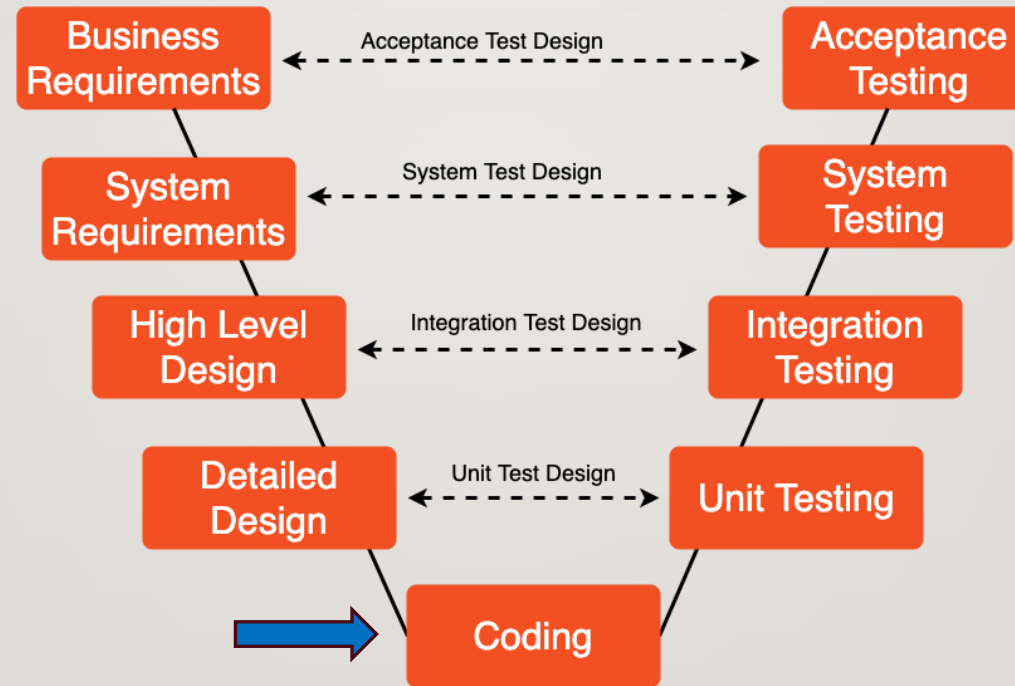
TEAM EXERCISE 9

Perform Detail Design of System Monitor Module using Process Flow Diagram

DETAIL DESIGN REVIEW BY EACH TEAM

Using the process flow diagram present team's design

SPRINT 3 – SYSTEM MONITOR



SDLC - V Model - notepub.io

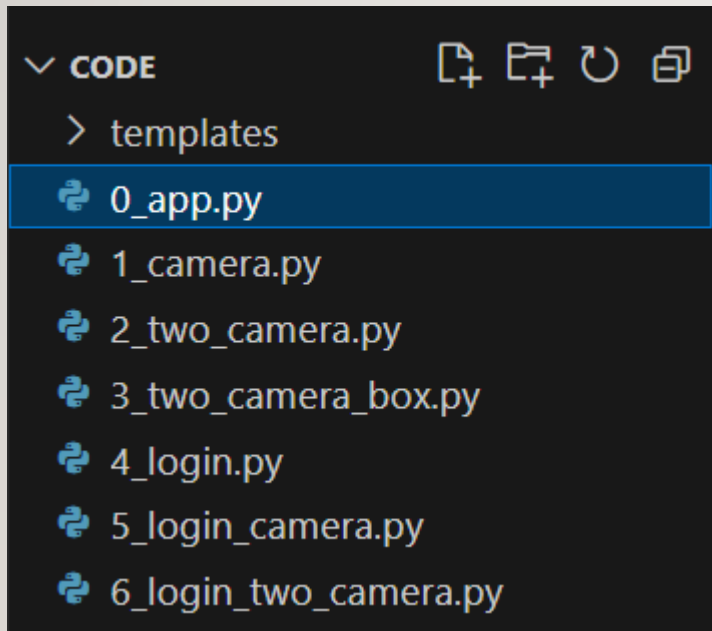
CODING HINTS

- Flask Basic Review
- SQLite Basic Review
- Webpage
 - Login page
 - Two video window
 - Alert Report
 - Status Captured and Following
- Database – SQLite
 - Login Data
 - Status Data

INTRODUCTION TO FLASK

- What is Flask?
A lightweight web framework for Python.
- Why Flask? Simple, flexible, good for beginners and small projects.
- `pip install Flask`
- `<project>/`
 - `|─ app.py` # Main Flask application file
 - `└─ templates/` # Folder for HTML templates
 - `└─── index.html`

FLASK HINTS



- HTML Reference:

[HTML elements reference - HTML: HyperText Markup Language | MDN](https://developer.mozilla.org/en-US/docs/Web/HTML/Element)

<https://developer.mozilla.org/en-US/docs/Web/HTML/Element>

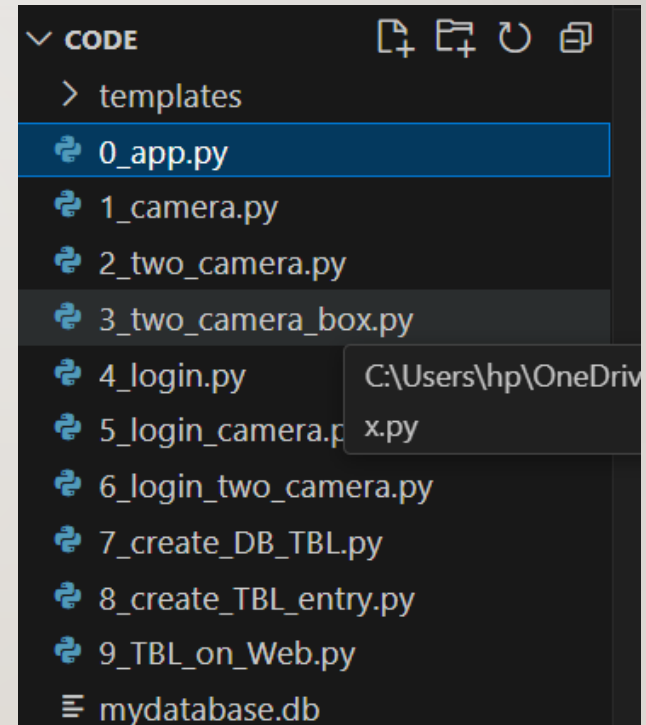
- CSS

[CSS: Cascading Style Sheets | MDN](https://developer.mozilla.org/en-US/docs/Web/CSS)

<https://developer.mozilla.org/en-US/docs/Web/CSS>

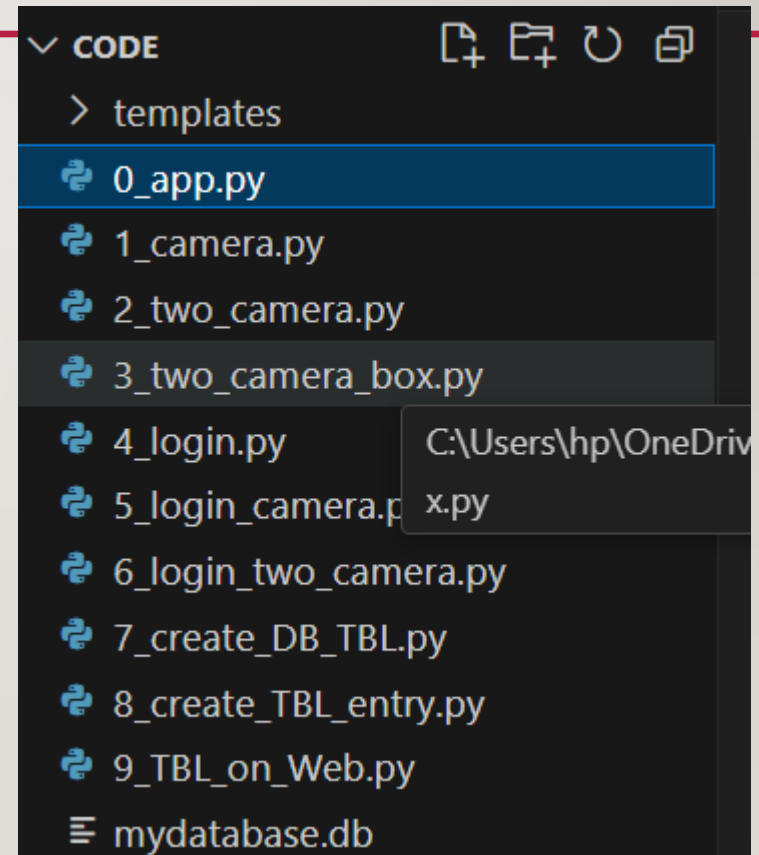
CODING HINTS

- Flask Basic Review
 - `sudo apt install sqlite3`
- SQLite Basic Review
 - SQLite is a lightweight, self-contained, serverless SQL database engine.



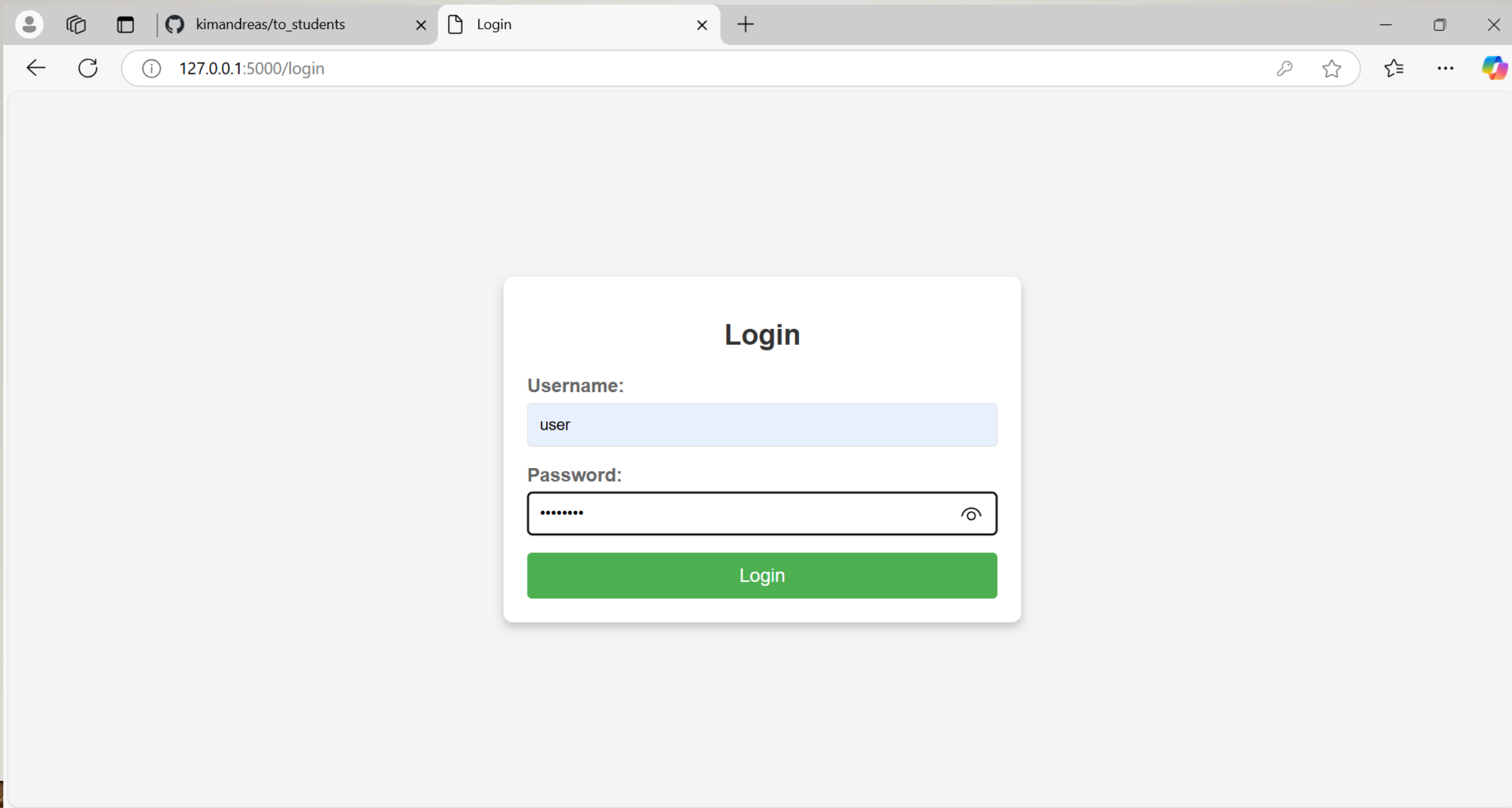
CODING HINTS

- Flask Basic Review
- SQLite Basic Review
- Webpage
 - Login page
 - Two video window
 - Alert Report
 - Status Captured and Following
- Database – SQLite
 - Detection Alert Data



CODING HINTS

- Database – SQLite viewer
 - `sudo apt install sqlitebrowser`
 - VSCode sqlite viewer extension



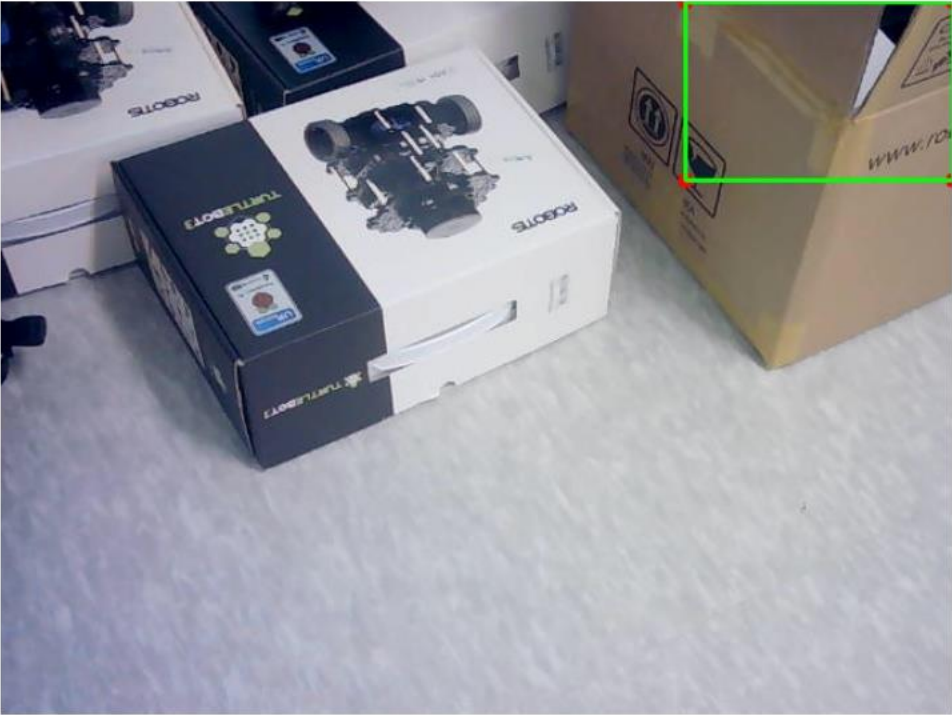
kimandreas/to_students


Welcome

127.0.0.1:5000/welcome

welcome, user!

You are now logged in.





Violations Detected

ID	Name	Date & Time
0	Truck	2024-11-06 10:30:22

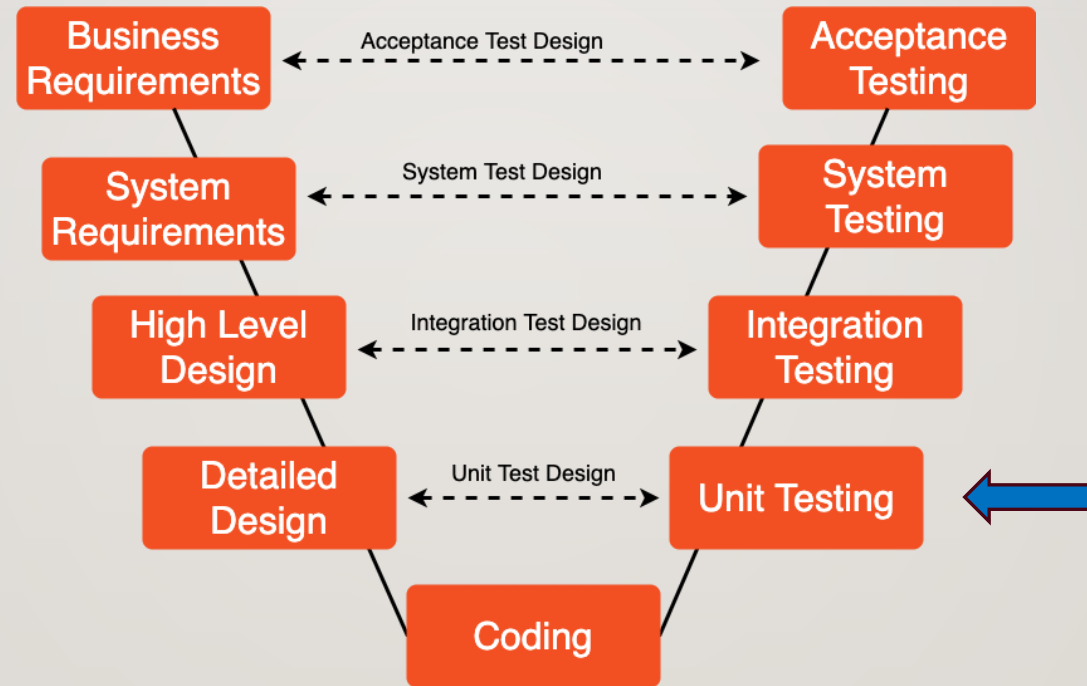
Track and Following

ID	Name	Date & Time
1	Dummy	2024-11-06 10:30:22

EXPECTED OUTCOME

- Sysmon with two windows and related Detection and tracking info
- ROS Nodes, Services, Topics

SPRINT 3 – SYSTEM MONITOR



SDLC - V Model - notepub.io

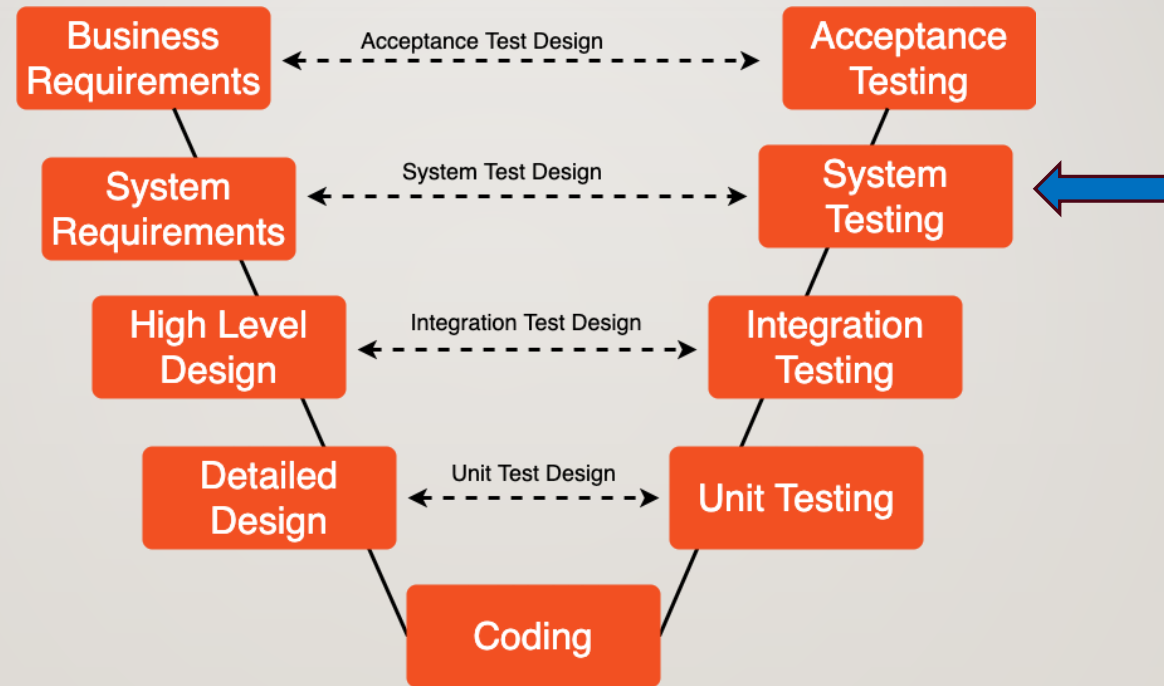
TEAM EXERCISE 10

Perform coding and testing of System Monitor Module

RESULTS & CODE REVIEW BY EACH TEAM

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

SYSTEM INTEGRATION & TEST



SDLC - V Model - notepub.io

TEAM EXERCISE II

Perform integration and testing of Detection, AMR Controller, and System Monitor Module

DEMONSTRATION OF SOLUTION BY EACH TEAM

Show actual results against the expected results and explain

프로젝트 RULE NUMBER ONE!!!

Are we still having
FUN!

