Final Presentation

TEAM_B

김준서
임승현
정종현
최지원

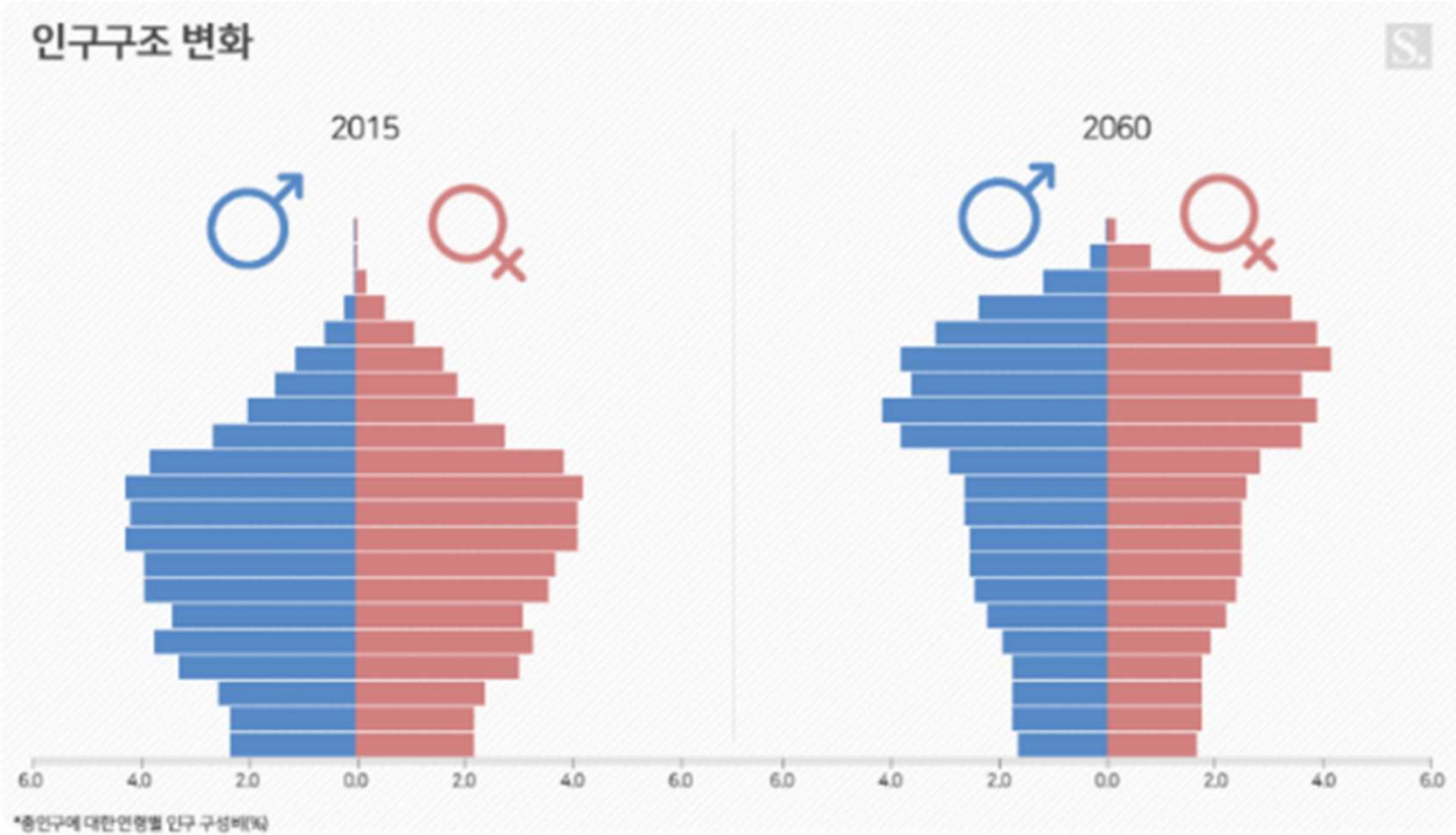
INDEX

1

Introduction

1 Introduction

1) Motivation

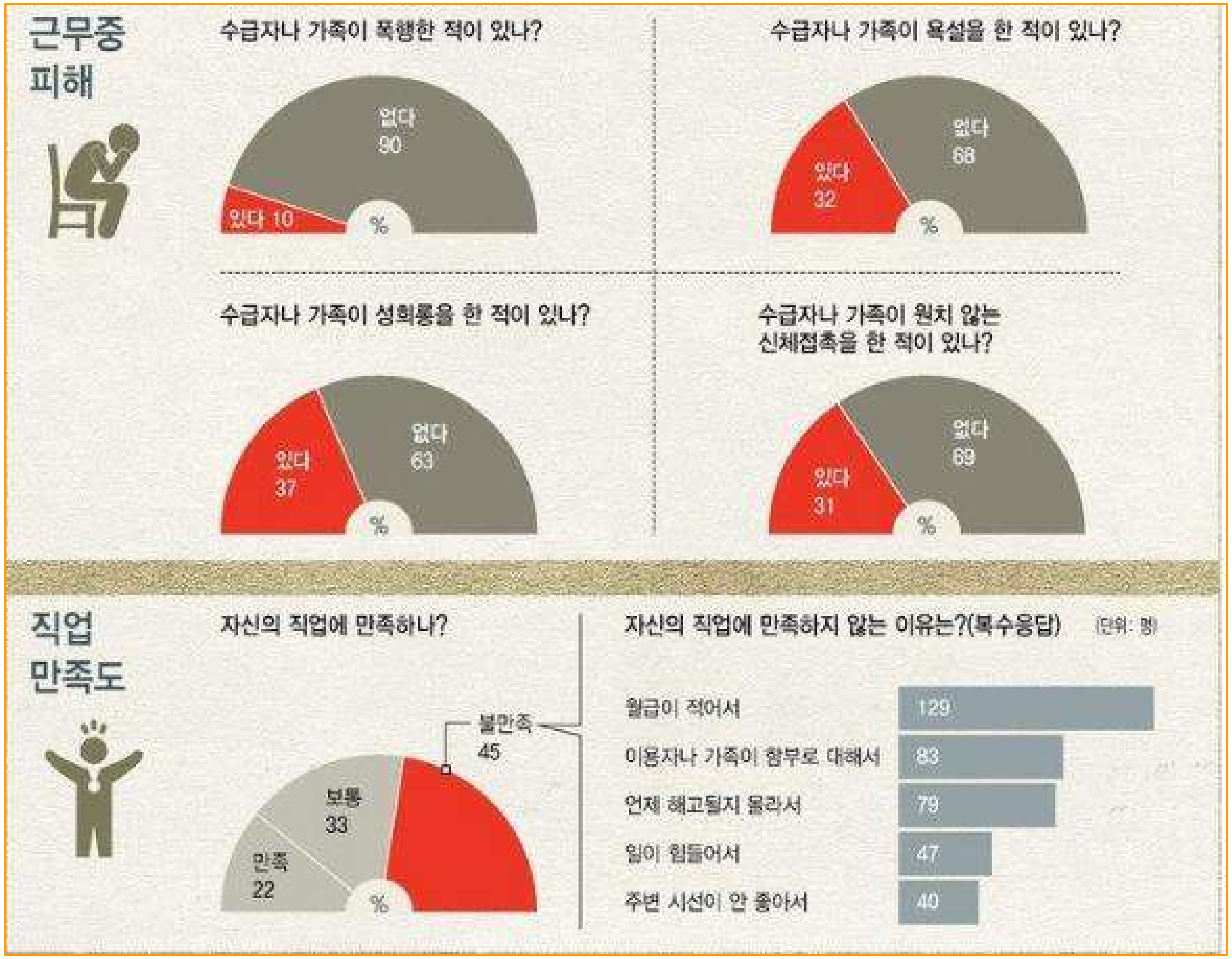


Increasing life expectancy

Increasing Sliver Industry

Decreasing birth rate

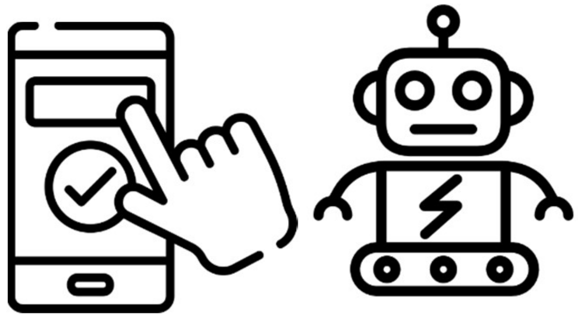
1) Motivation



Bad Working Condition

1 Introduction

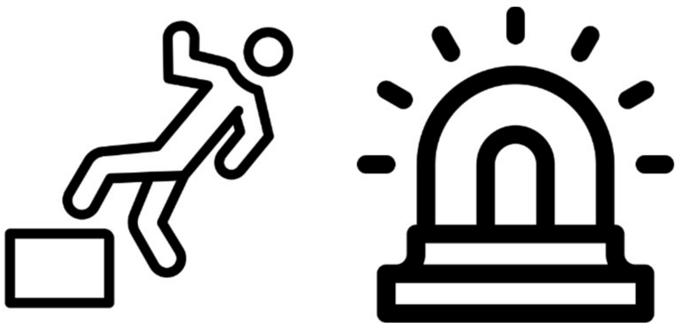
2) Objective



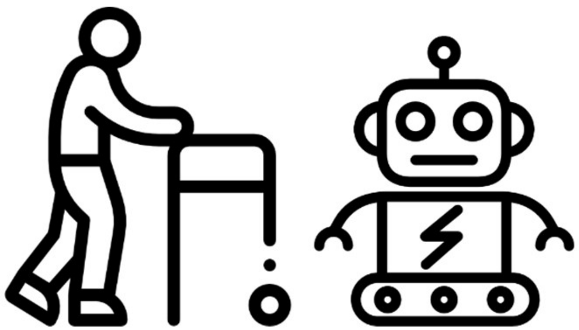
1. Robot Call on User Request

When user call the robot, robot will move to designated area.

2. Emergency Detection



When robot detects emergency situation (fall accident),
It takes picture and send alarm to nursery



3. Stroll Assistant

User can take a walk with robot without fear of get lost.

2

Project Progress

Role of each member

- **Role of each member:**

김준서: Layout and functions about App, BackEnd about App, Alarm of Emergency

임승현: Computer Vision in Robotics and Implement about Navigation(DL)

정종현: Build Robot’s Operating Environment, Server and BackEnd about Robots

최지훤: Single Page Application, Basic Layout and functions, Socket in FrontEnd

2 Project Progress

2) Schedule with milestones

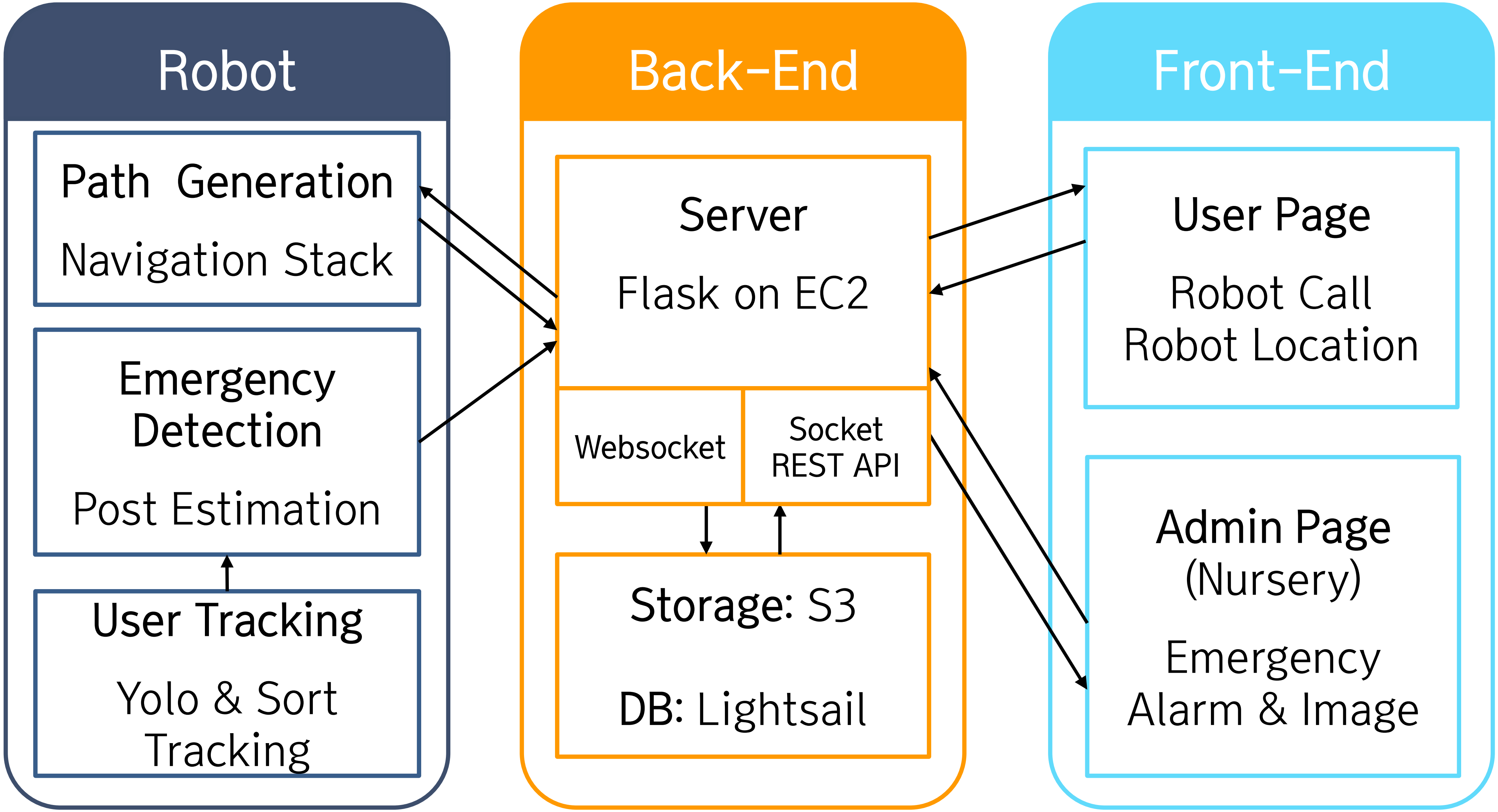
Part	Function	Week6	Week7	Week8	Week9	Week10	Week11	Week12	Week13	Week14	Week15
	로봇 시뮬레이션 설정										
	로봇 센서 설정										
Robotics	로봇 Navigation 설정										
	로봇 actor plugin 개발										
	서버 통신 용 service 작성										
	넘어짐 감지 구현										
Deep learning	Depth yolo 구현										
	sort-track 주행 알고리즘										
	서버와 연결 및 서비스 생성										
	layout 디자인 및 구현										
	flow chart 작성										
	map 좌표입력 기능 구현										
	rest api test										
Front End	로봇 호출 및 산책기능 구현										
	websocket api 적용										
	User, Admin 분리										
	Notification in FrontEnd										
	application architecutre 수정										
	Database/Storage 연결										
	로봇 Emergency 상황 정보 저장										
	로봇 서버 연결										
	목적지 API										
	Login Register in app										
Back End	react-flask 연동										
	notification in BackEnd										
	react-flask 소켓통신										
	로봇 위치 전달										
	관리자 앱 응급상황 목록 api 작성										
	로봇 통신 용 service 노드 작성										

3

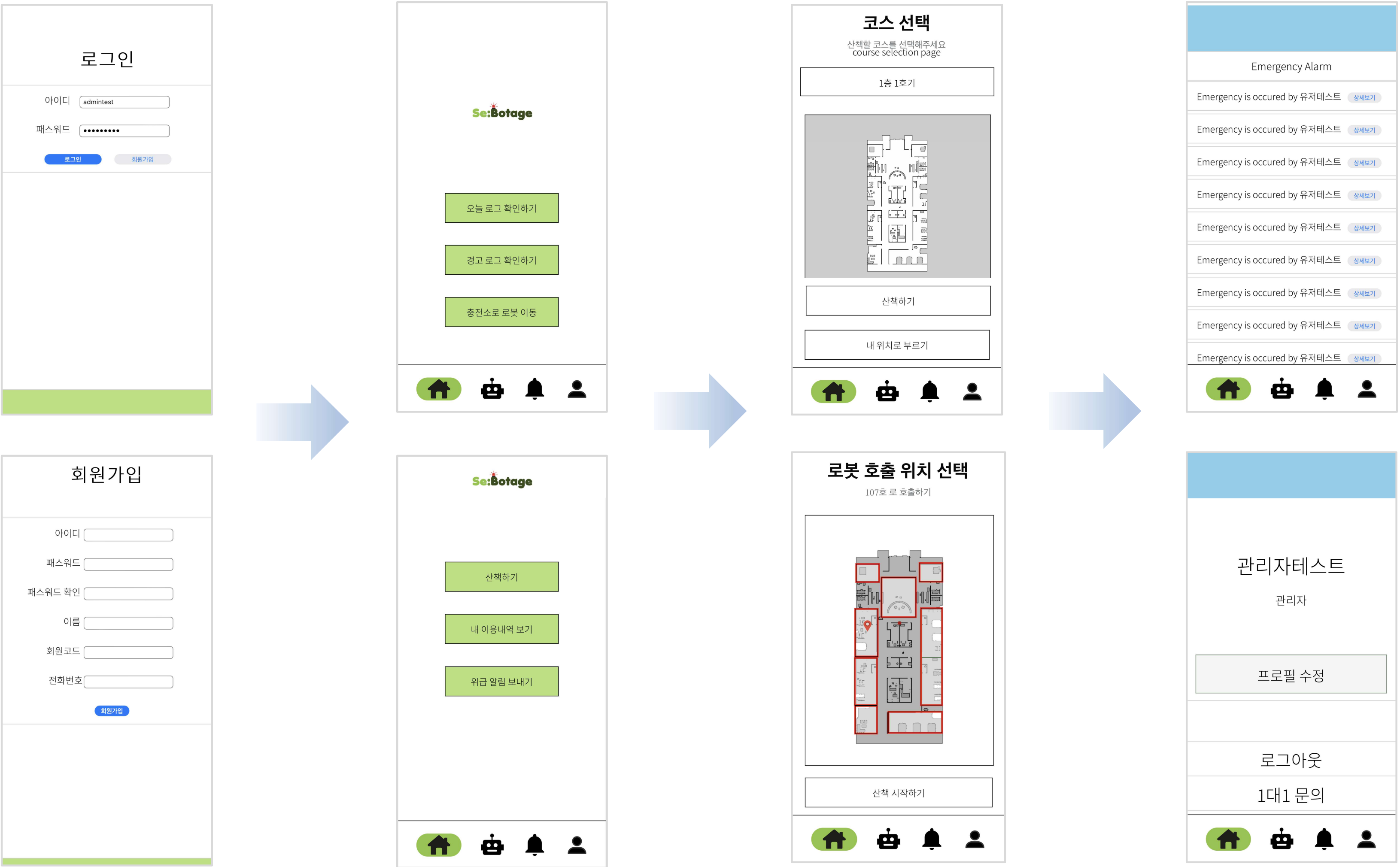
Design & Implementation

3 Design & Implementation

1) Architecture



2) App Design

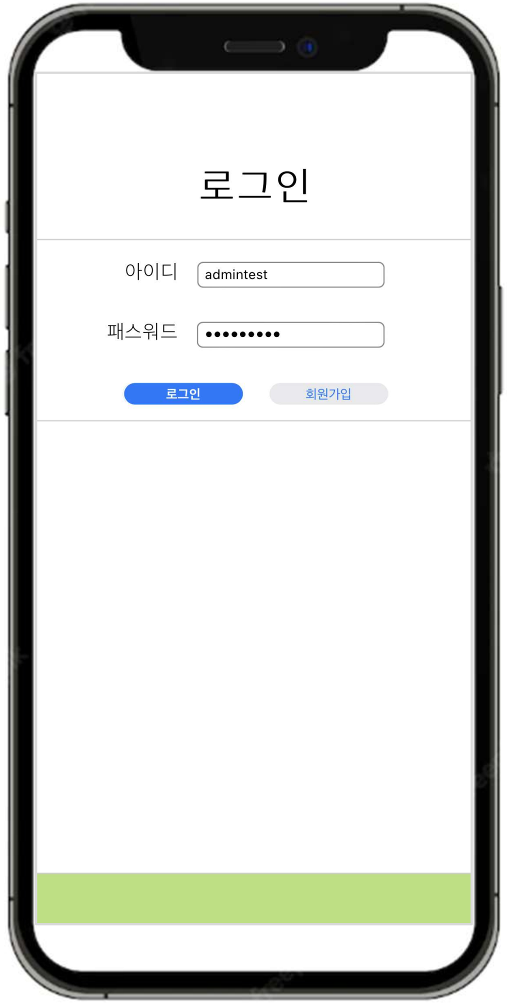


로그인
아이디
admintest
패스워드
로그인
회원가입
회원가입
아이디
패스워드
패스워드 확인
이름
회원코드
전화번호
회원가입
Se:Botage
오늘 로그 확인하기
경고 로그 확인하기
충전소로 로봇 이동
Se:Botage
산책하기
내 이용내역 보기
위급 알림 보내기
코스 선택
산책할 코스를 선택해주세요
course selection page
1층 1호기
산책하기
내 위치로 부르기
로봇 호출 위치 선택
107호 로 호출하기
산책 시작하기
Emergency Alarm
Emergency is occured by 유저테스트
관리자테스트
관리자
프로필 수정
로그아웃
1대1 문의

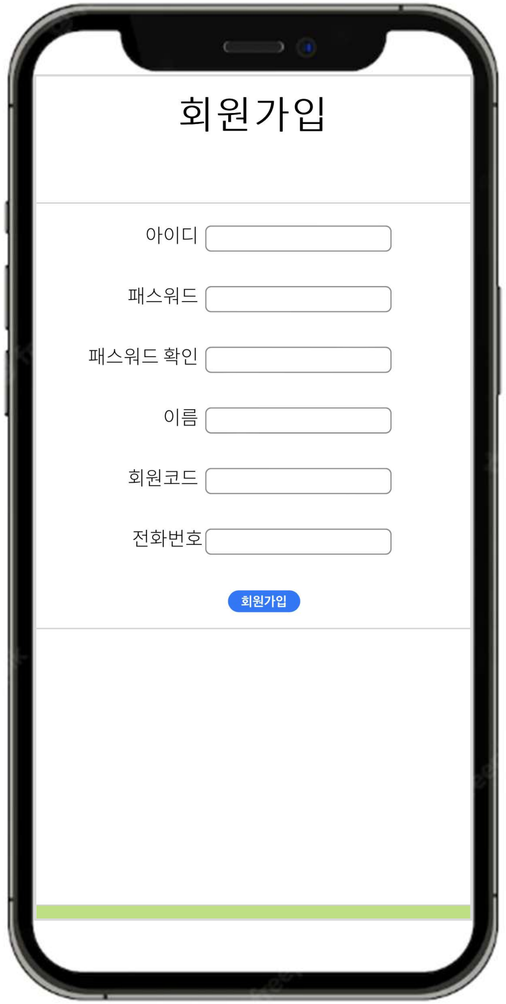
3 Design & Implementation

2) App Design – **Login & Register**

Login



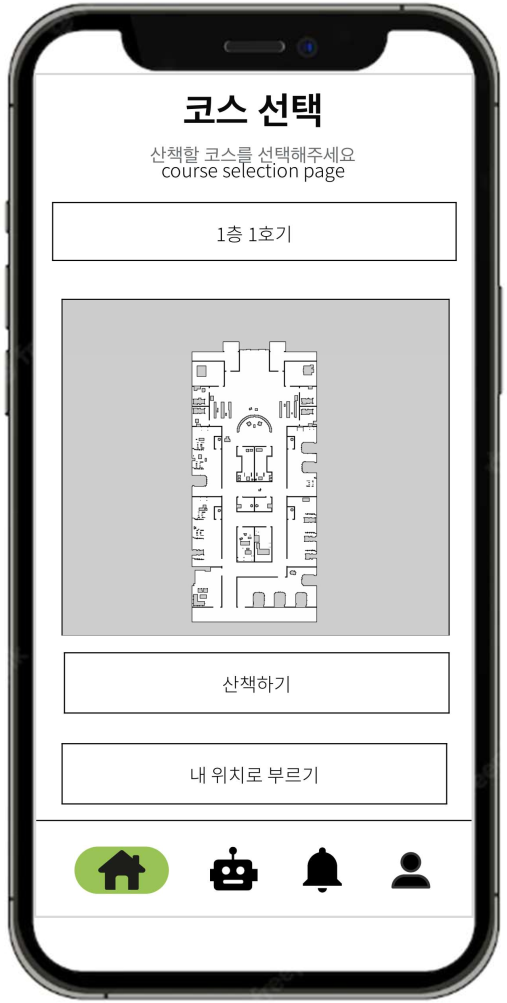
Register



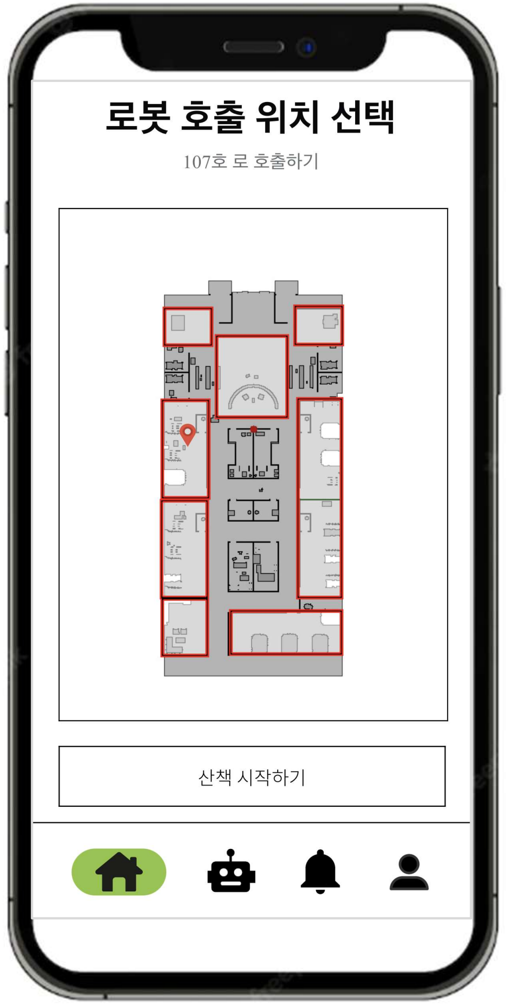
3 Design & Implementation

2) App Design - **Walking**

Main

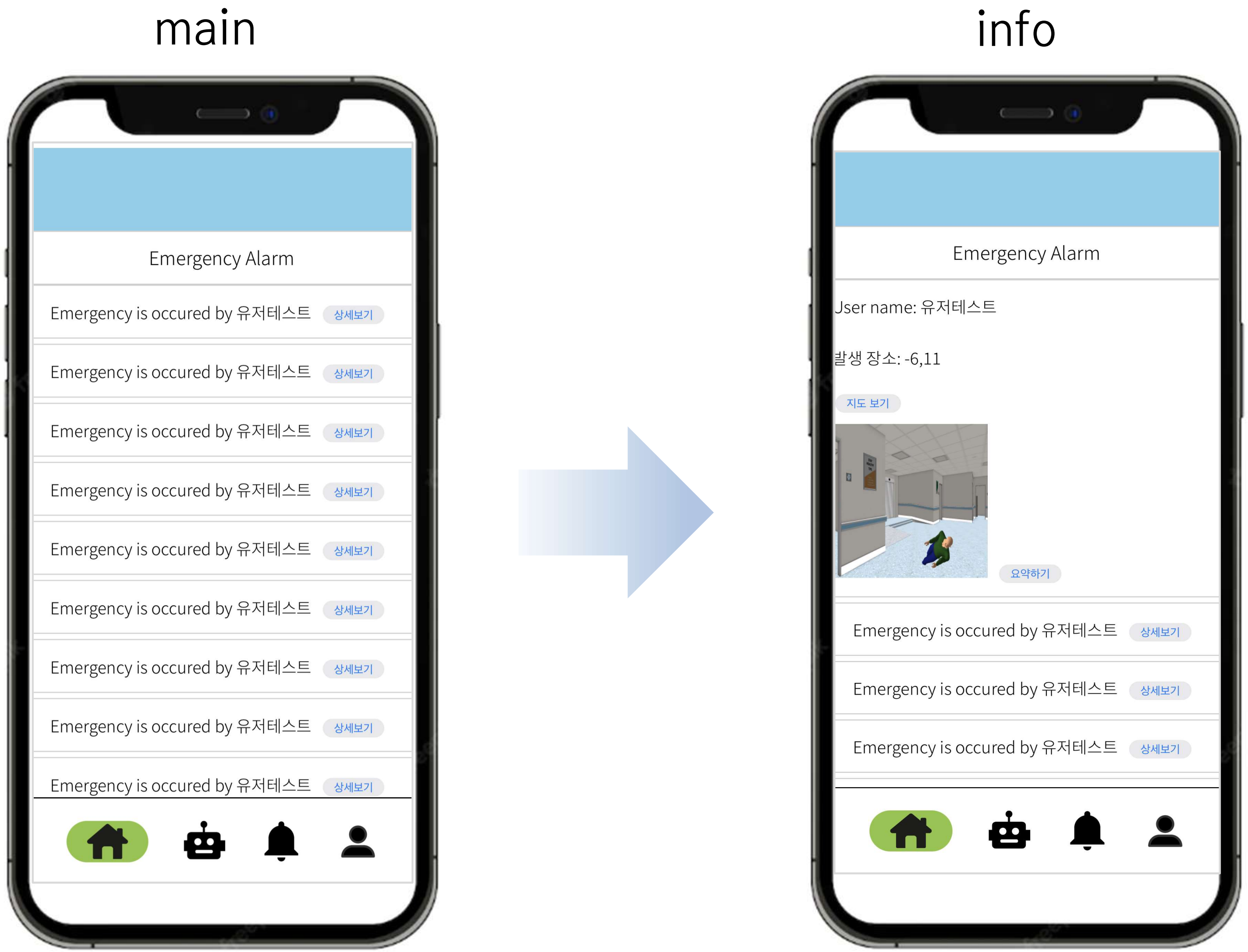


Select course



3 Design & Implementation

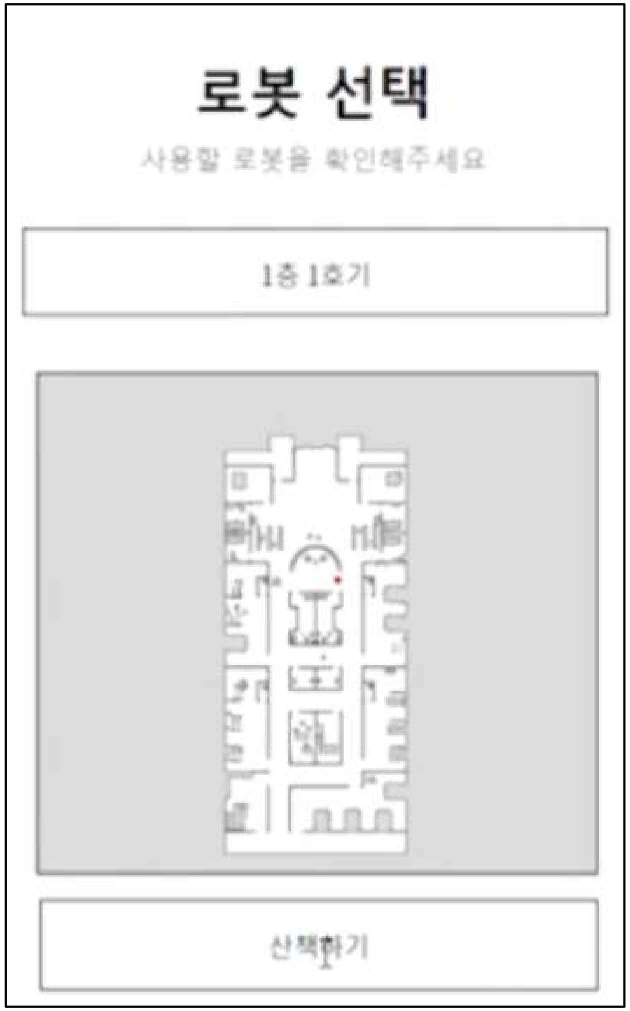
2) App Design - **Notification**



3 Design & Implementation

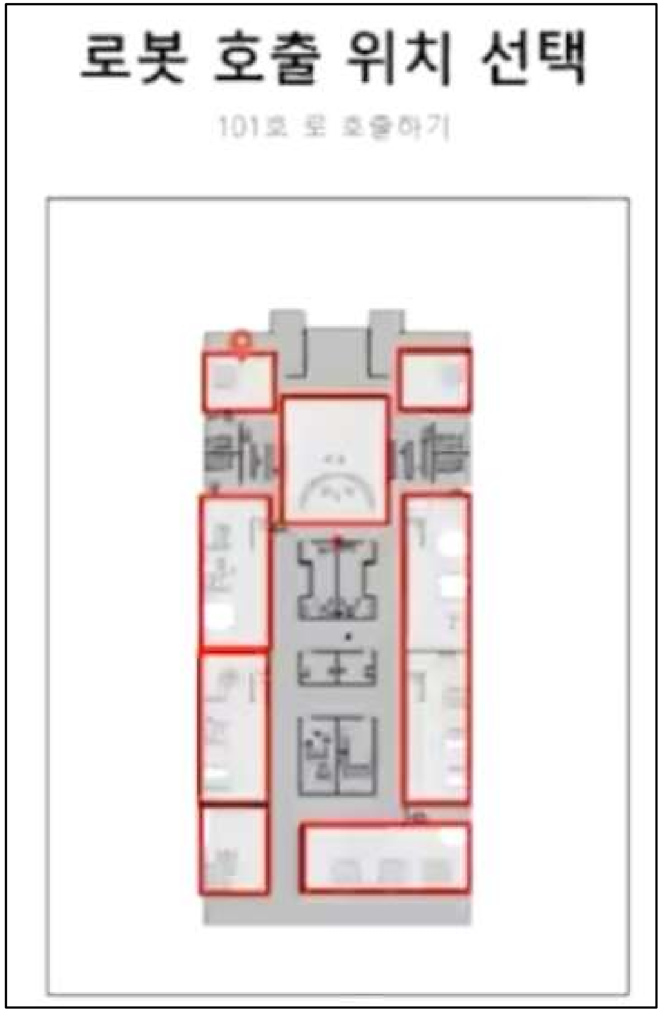
3-1) Implementation: **Call Robot**

Start stroll assistance



3 Design & Implementation

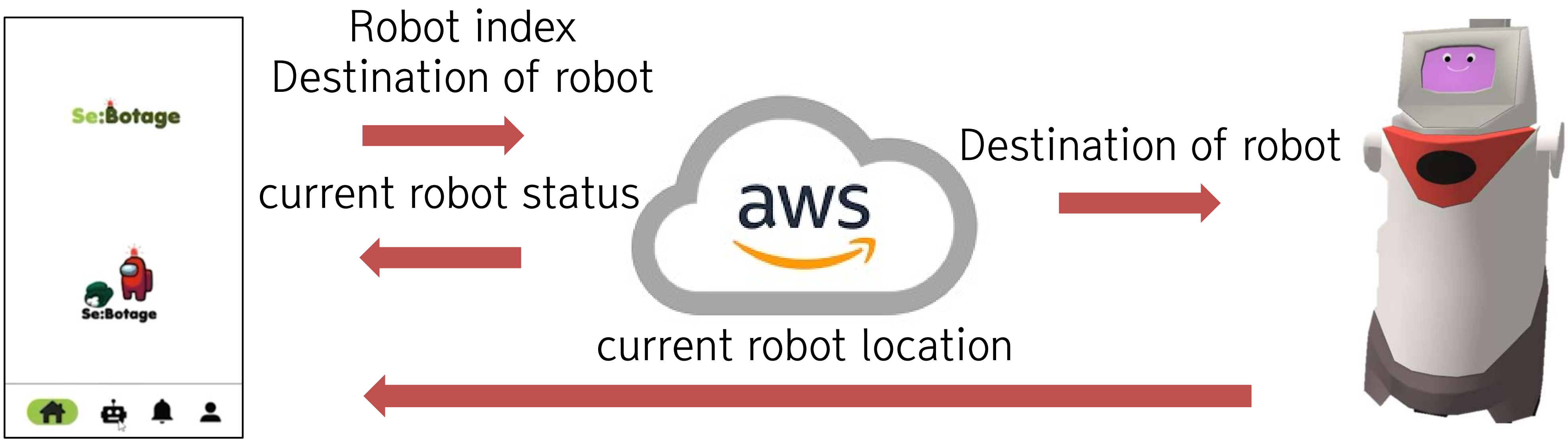
3-1) Implementation: **Call Robot**



Call robot to desired location

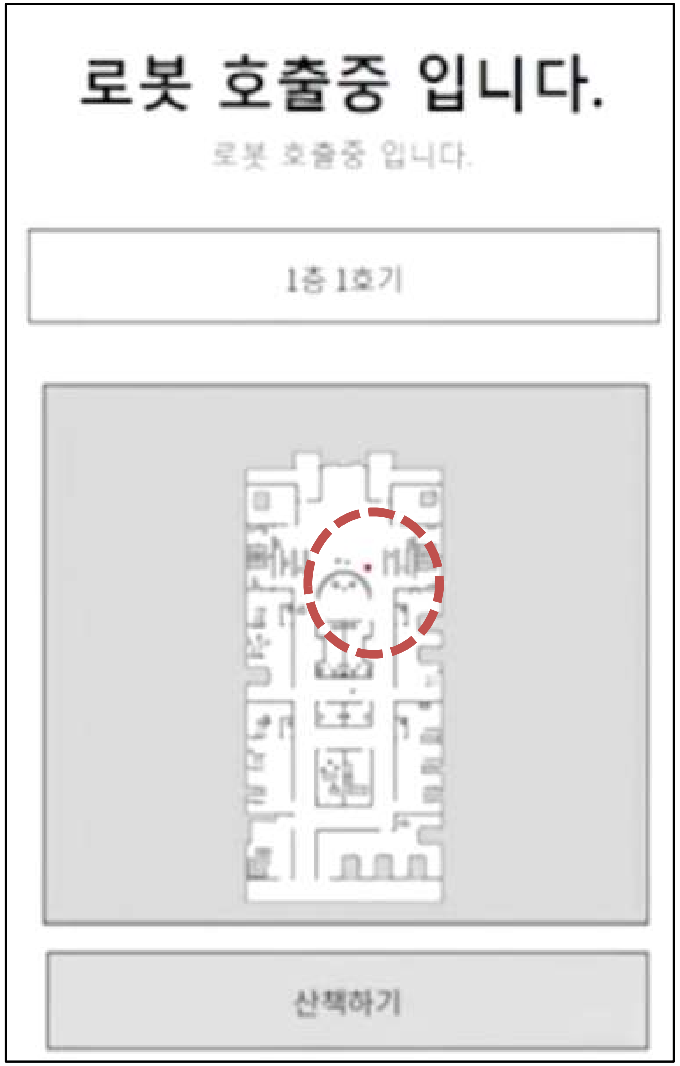
3 Design & Implementation

3-1) Implementation: **Call Robot**

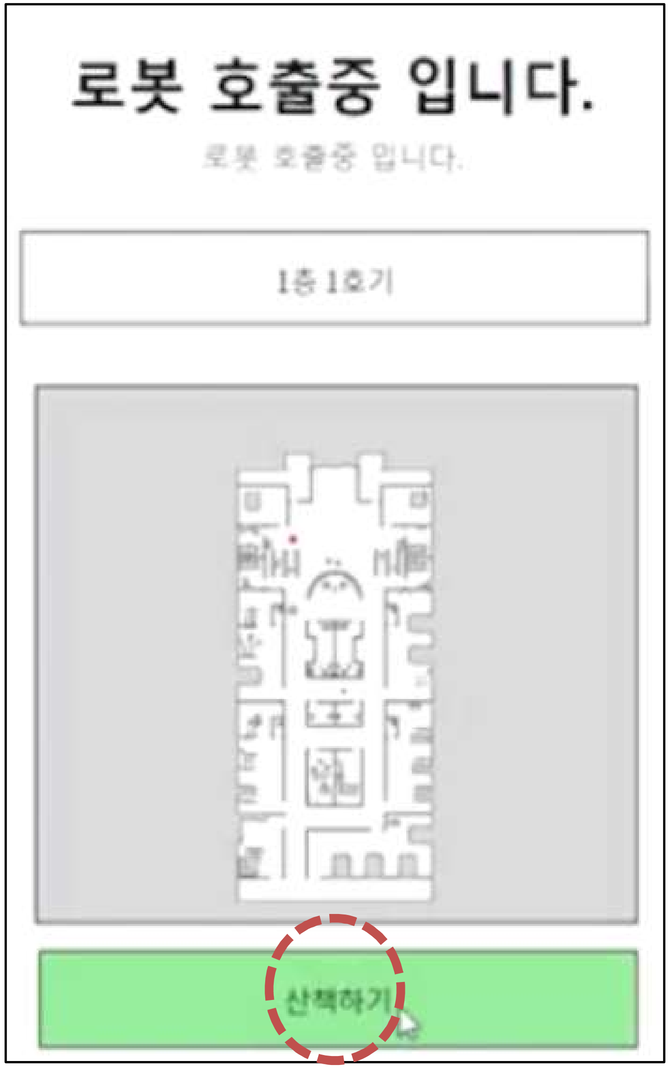


3-1) Implementation: **Call Robot**

Update robot location



Robot arrived & button activated



3 Design & Implementation

3-1) Implementation: **Call Robot**

Demo video

3 Design & Implementation

3-2) Implementation : **Emergency Detection**

Emergency Detection : Detecting "Fall-off" motion

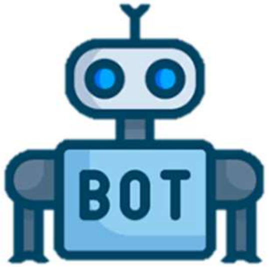
Method: Openpose (Finding all "keypoints" of humans, ex) head, elbows...)

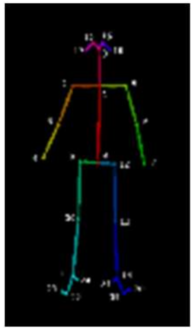
3 Design & Implementation

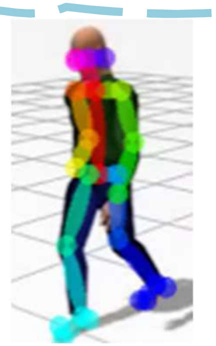
3-2) Implementation : **Emergency Detection**

Openpose : Use Body25 model, 25 keypoints of the human body

Dataset : Coco and MPII (pretrained)







Robot Camera

/image_raw
(ROS Topic)

Openpose node
(python)

Image process

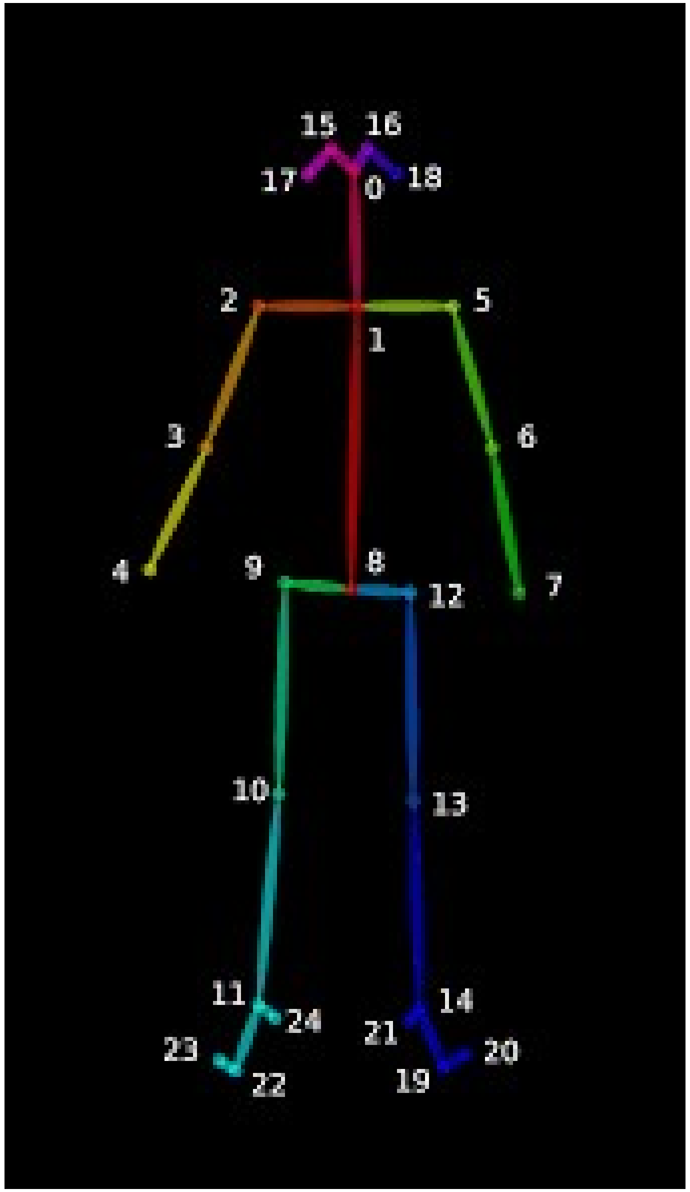
Keypoints
(Array type)

3-2) Implementation : **Emergency Detection**

"Fall-off" detection using "keypoints"

We use **head(point 1)** , **waist(point 8)** , **and foot (point 22 or 19)**

Assumption : Both robot and a user are on the **flat surface** (no slope)



3-2) Implementation : **Emergency Detection**

Key Algorithm

1) A full body of a user should be visible (from head to toe).

2) A head should be under certain pixel. ($y_{head} \leq head\_threshold$)

3-1) The line connecting the points of **head and waist** should be under certain angle with the floor

$(\text{lower\_bound} \leq \tan^{-1}(line\ with\ head\ \&\ waist) \leq upper\_bound)$

3-2) The line connecting the points of **waist and foot** should be under certain angle with the floor

$(\text{lower\_bound} \leq \tan^{-1}(line\ with\ head\ \&\ waist) \leq upper\_bound$

You should meet *1, 2 , 3-1* or *1 , 2 , 3-2*

3-2) Implementation : **Emergency Detection**

Whole representation of the Fall-off-Detector algorithm
Return *True* when the user fell off, otherwise *False*

Algorithm 1 Algorithm for Fall-off Detector

Require: $head\_threshold$, min_angle , max_angle
  procedure FALL-OFF-DETECTOR($head\_point$, $waist\_point$, $foot\_point$)
    if Either $head\_point$ or $waist\_point$ or $foot\_point$ are $None$ **then**
      return $False$
    end if
    if $head\_point.y \leq head\_threshold$ **then**
      $upper\_body\_slope = (head\_point.y - waist\_point.y)/(head\_point.x - waist\_point.x)$
      $upper\_body\_angle = \tan^{-1}(upper\_body\_slope)$
      Compute $lower\_body\_angle$ in the same manner using $waist\_point$ and $foot\_point$
      if $(min\_angle \leq upper\_body\_angle \leq max\_angle)$ or $(min\_angle \leq lower\_body\_angle \leq max\_angle)$ **then**
      return $True$
      end if
    end if
      return $False$
  end procedure

3 Design & Implementation

3-2) Implementation : **Emergency Detection**

It also works on real images!

We will discuss *Limitations* later on ......

3-2) Implementation : **Emergency Detection**

Emergency Notification

3-2) Implementation : **Emergency Detection**

Demo video of Emergency detection

3-3) Functional Implementation : Outdoor Strolling Service

Description

Stroll in outdoor with our “user”, checking if user follow the robot in real-time

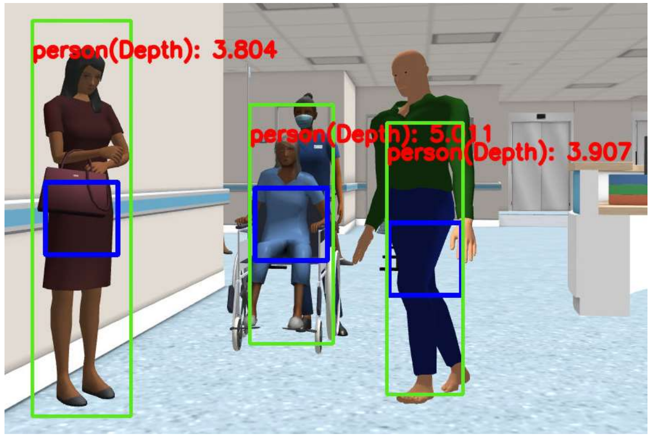
Implementation (Our own contributions)

1) Depth-Yolo : Detecting object and **depth data** (How far a people locates)

2) Sort-track Algorithm : Our own algorithm to track our user between multiple humans

3-3) Functional Implementation : Outdoor Strolling Service

Depth-Yolo

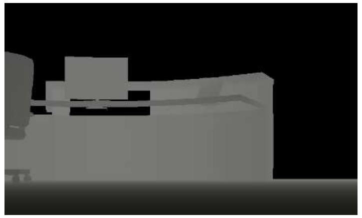


Yolov3

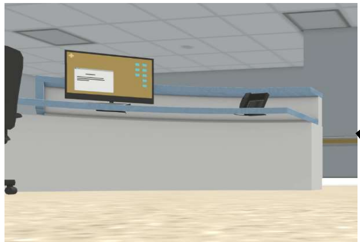
Coco dataset to detect “Person”

Use “tiny_weight” : For real-time speed

/depth/image_raw



/rgb/image_raw



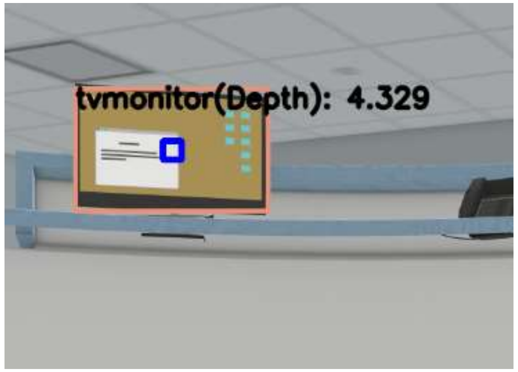
Decode

OpenCV

(Height, Width)
Value : Depth (meter)

(Height, Width,3)
Value : RGB (0,255)

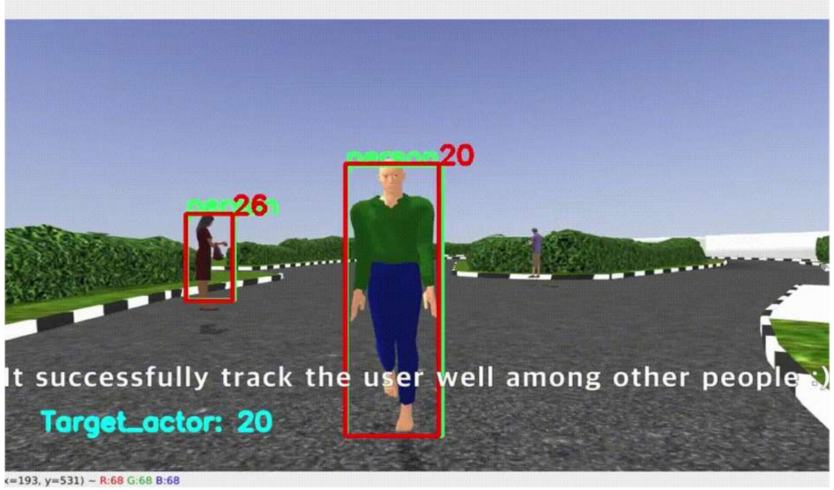
YOLO

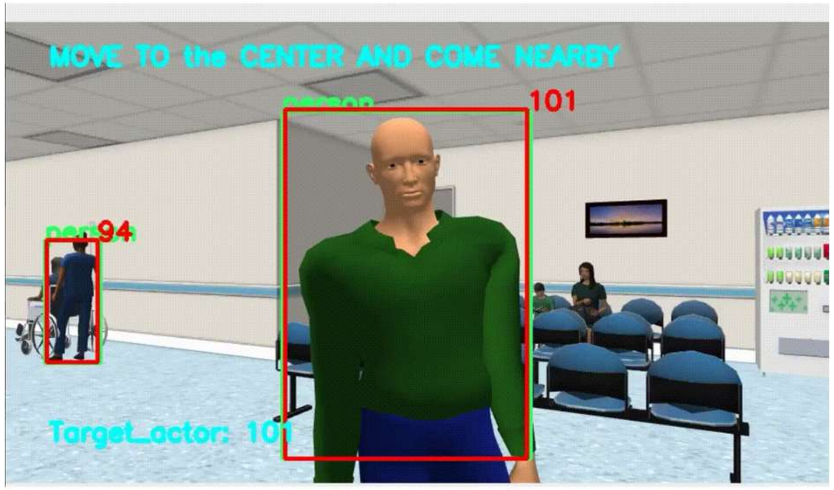


3 Design & Implementation

3-3) Functional Implementation : Outdoor Strolling Service

About Sort track





To identify "users" between multiple humans.

Adding "indices" to each Yolo bounding boxes(bbox).

Input : bbox (of humans) from our yolo node

Output : bbox with **indices** (Left image)

3 Design & Implementation

3-3) Functional Implementation : Outdoor Strolling Service

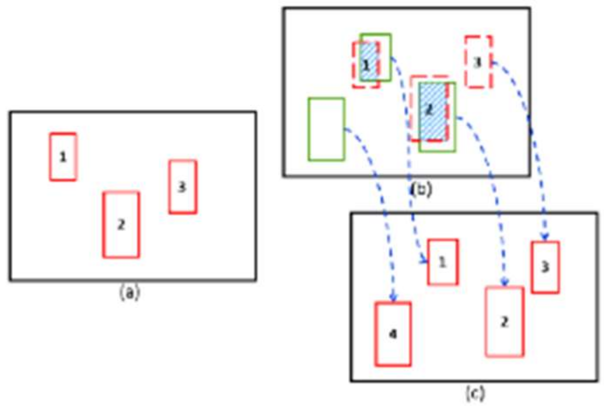
Key Algorithm of Sort-track

Kalman filter : Predict **where the bbox will be** based on current velocity and location

Hungarian Algorithm : **Match** the “Predicted bbox” from Kalman filter with “Actual bbox”

cf) Deep-sort : Add DNN to enhance the velocity and location estimation (Help the role of Kalman filter)

We use deep-sort in demo, pretrained weight.



Hungarian Algorithm in sort track

3 Design & Implementation

3-3) Functional Implementation : Outdoor Strolling Service

Our Implementation

Depth yolo + Sort track . . . We are able to know **who is our "user"** and **how far is him.**

It consists of two status : Tracking(idle), (Re)Identification

Tracking

Identification

Cannot find "target_actor" or too far away

Store index of "target_actor" after identification

Keep track of a users and measuring depths.

Designate the nearest person in front of camera as a "User"

3 Design & Implementation

3-3) Functional Implementation : Outdoor Strolling Service

Demo video of Outdoor Strolling Service

4

Challenges

4 Challenges

1) Challenges in Robotics & DL

1. How to implement "Fall-off detector" with the output of human keypoints in openpose?

Test again and again in our simulated environments to revise algorithms. (Heuristic approach)

We also had to find best parameters that best fits in our environments.

After feedback of our classmates, we also test on real videos.

2. How to distinguish our "User" from others?

This function didn't mention in proposal, but we thought it is essential for real usage.

First trial: Face recognition -> Failed!

We thought it is inappropriate for **simulated humans**, features cannot well distinguished.

Second trial: Sort tracking -> Success!

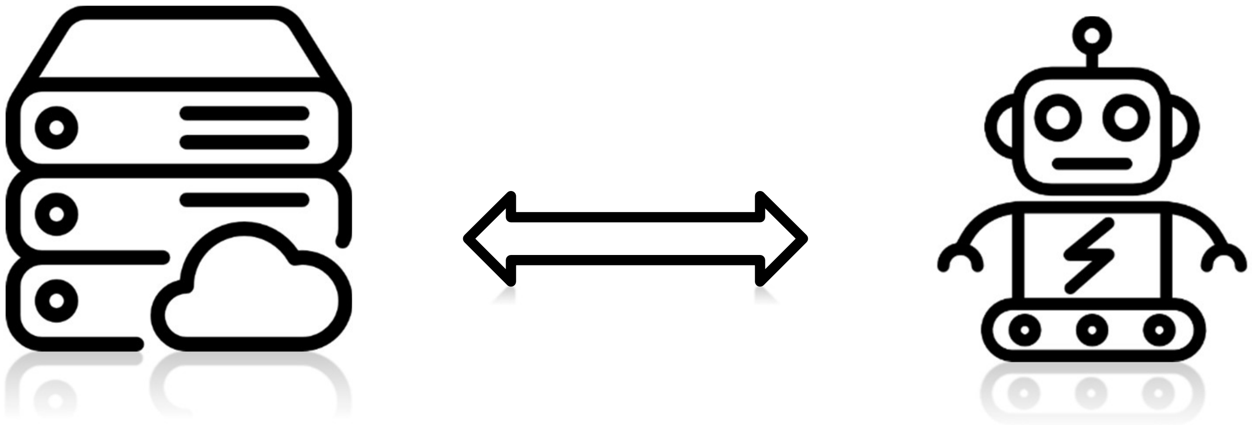
How should we handle when index of our user changed due to error or temporally "out-of-sight"?

Make Two status : Tracking & Identification, able to move slow and re-identify our users smartly.

4 Challenges

2) Server

a) Connection between Server and Robot



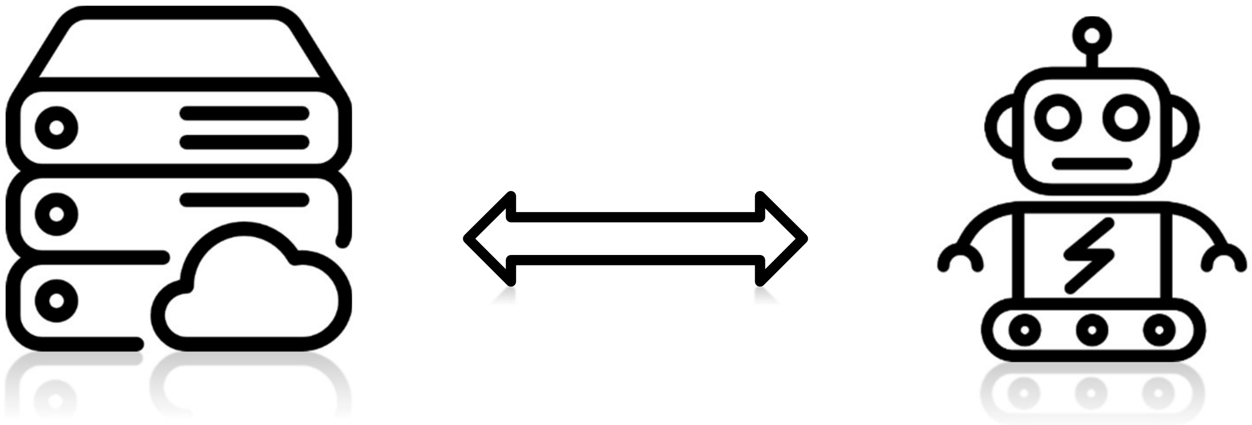
Security Issue

ROS TCP Communication → Websocket Communication (roslibpy)

4 Challenges

2) Server

b) Overhead Optimization



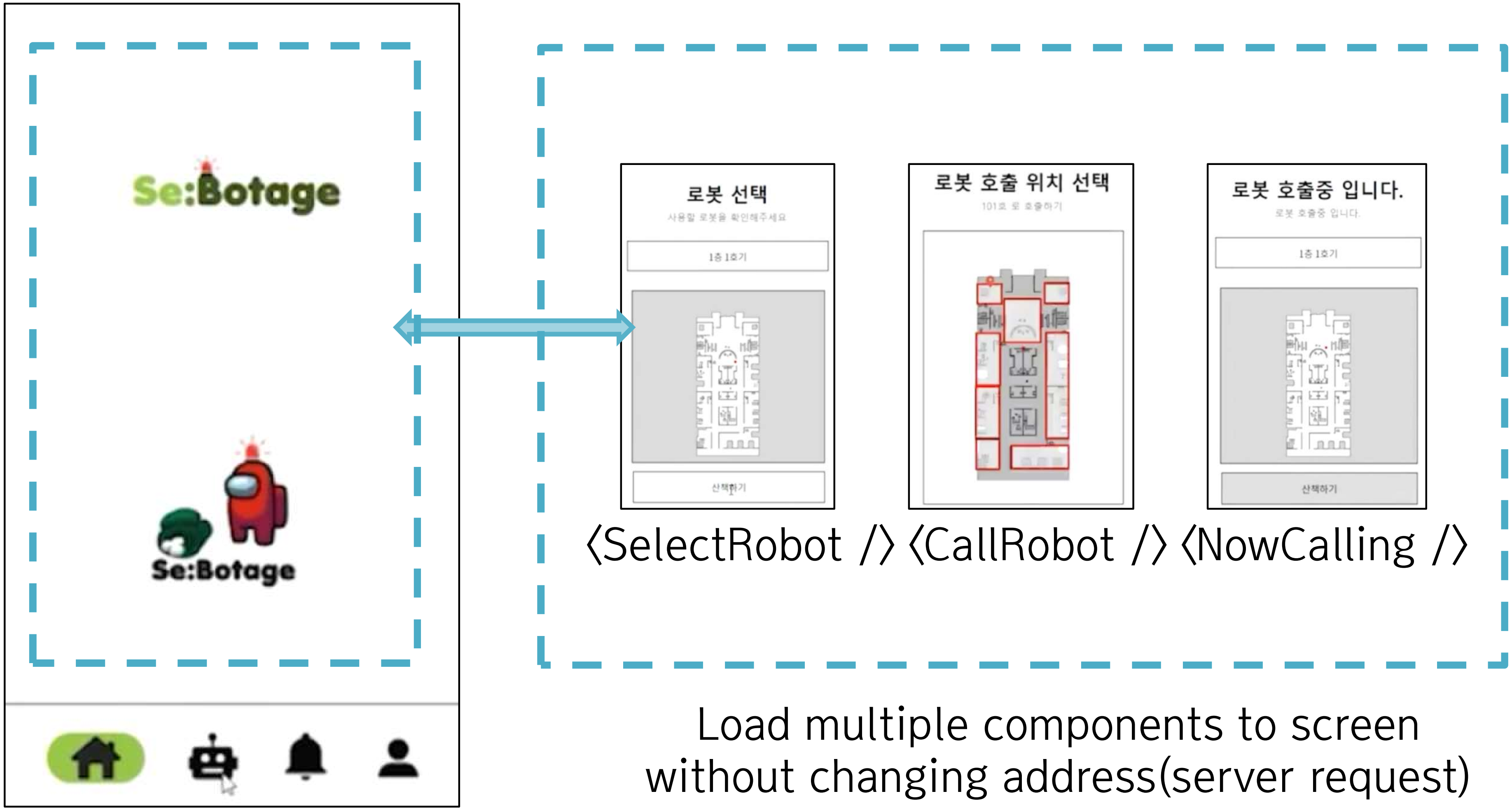
Overhead Issue

ROS Topic Communication (Subscriber/Publisher) → ROS Service Communication (Request/Response)

4 Challenges

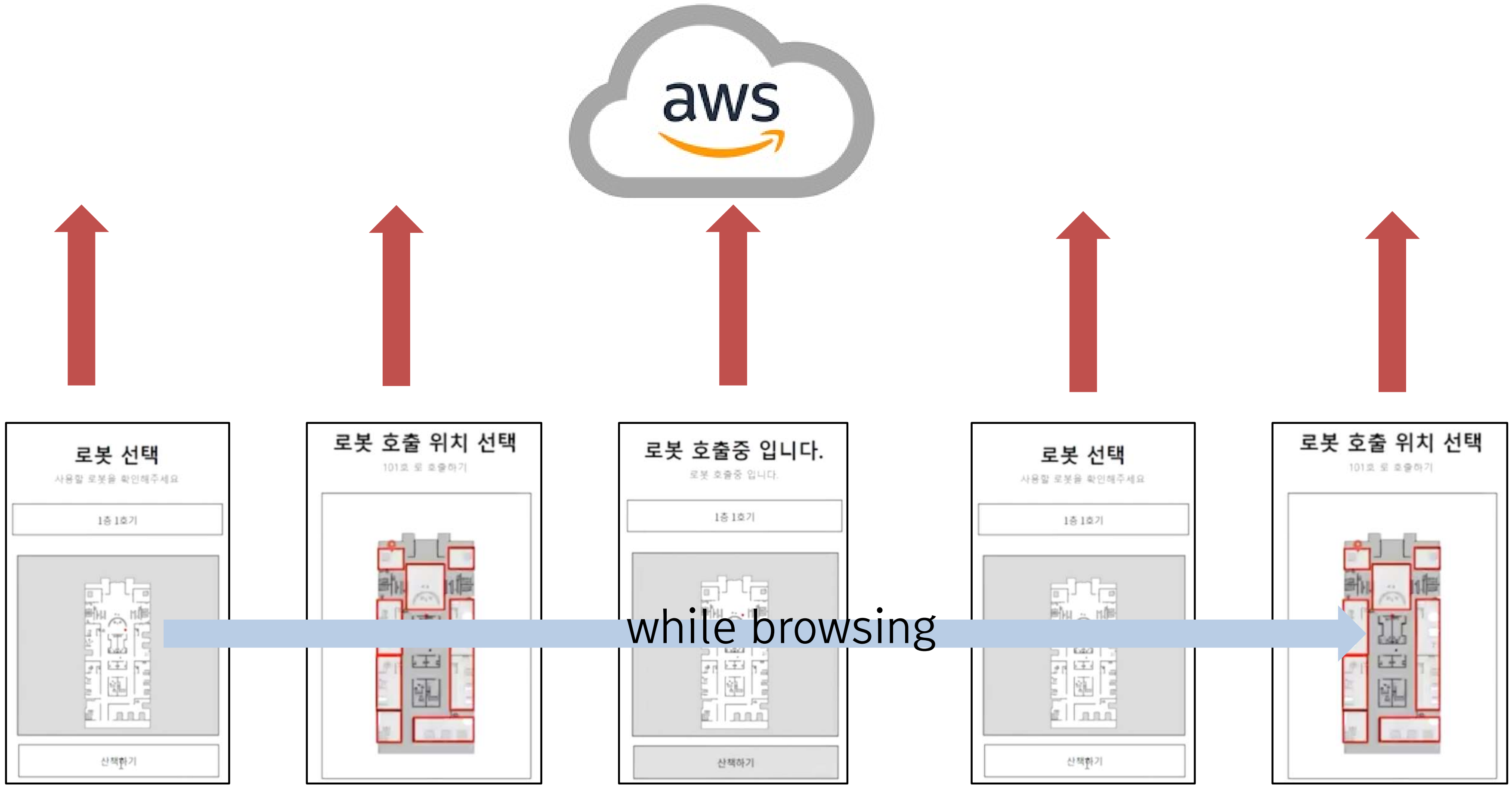
- 3) Frontend – app architecture

Single address: 111.222.33.44:5000/start



Load multiple components to screen
without changing address(server request)

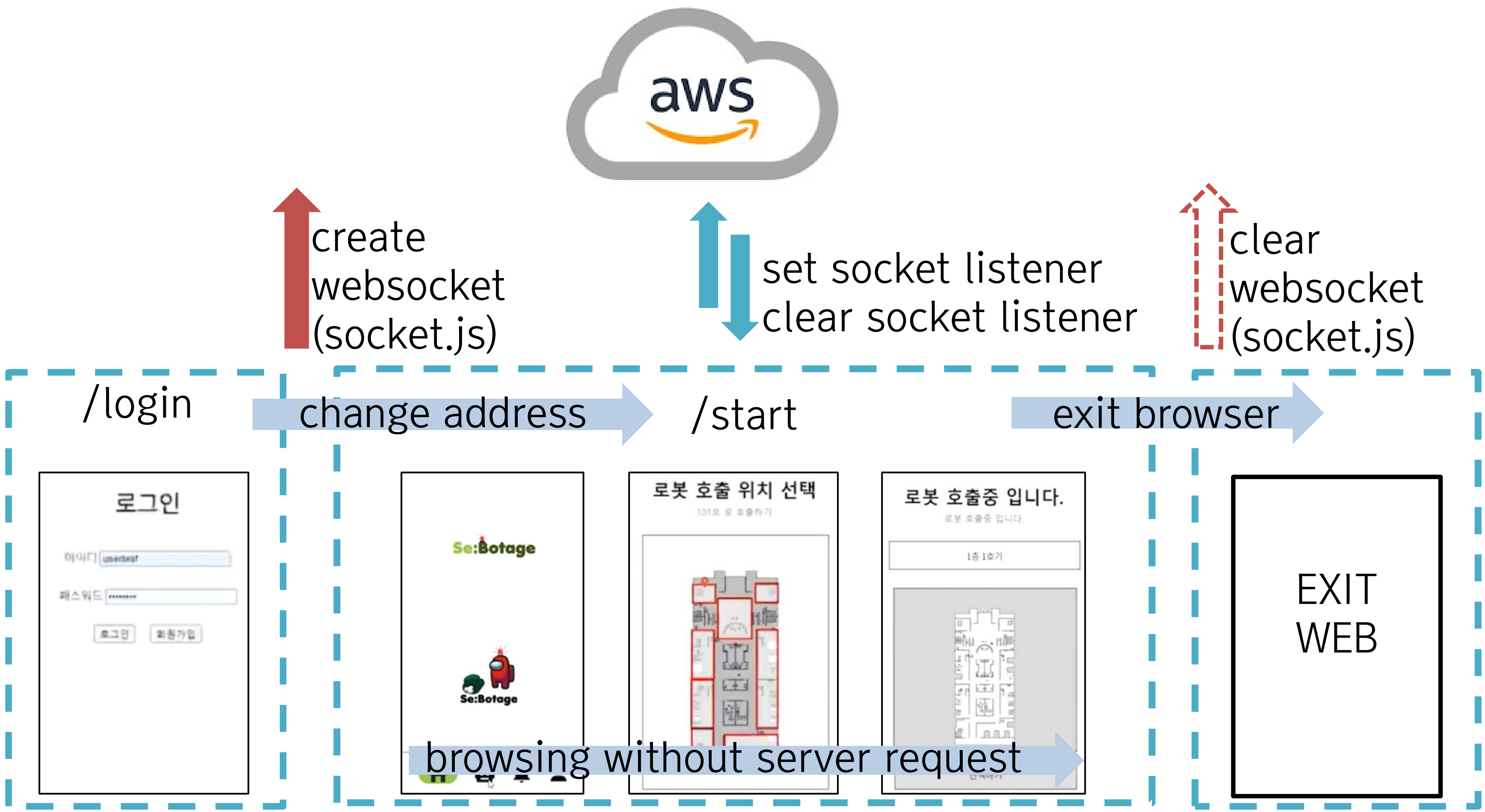
3) Frontend – app architecture



Every single components create websocket channel

4 Challenges

- 3) Frontend – app architecture

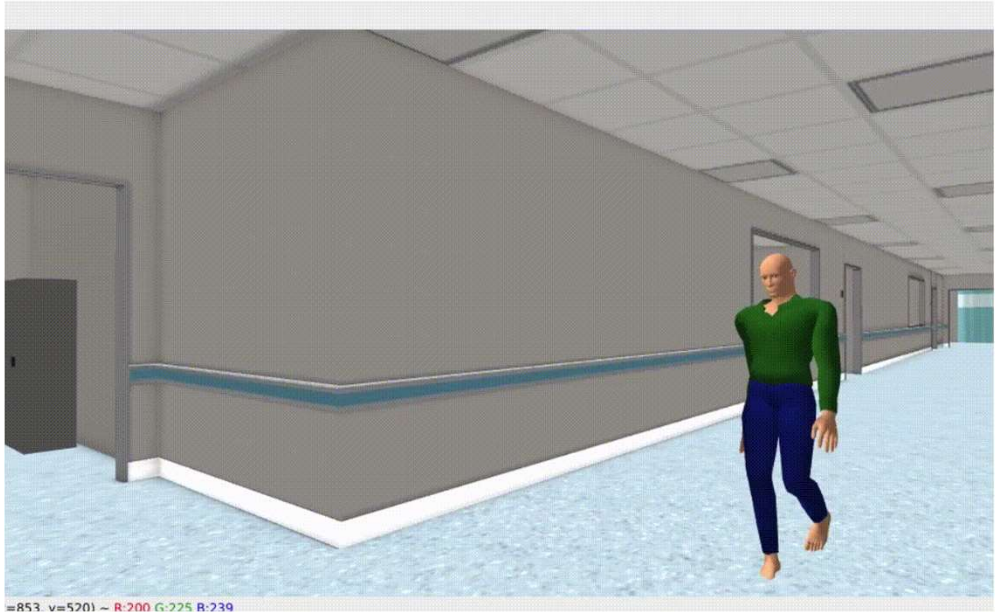


5

Limitation

5 Limitation

1) Simulation



Awkward walking of actor

Similar face of human in simulation

5 Limitation

2) Fall-off detector in emergency service

1) A robot should be on the flat surface (slope might affect "thresholds" of y axis)

2) Only taking account of a static image not a sequence . . .

-〉 Cannot judge whether a player is fell off or "just lying on the floor"

3) Other possibilities : Error in Openpose(pose estimation), and parameters tuning needed in different environments ... etc.

5 Limitation

3) Sort tracking in Outdoor Strolling Service

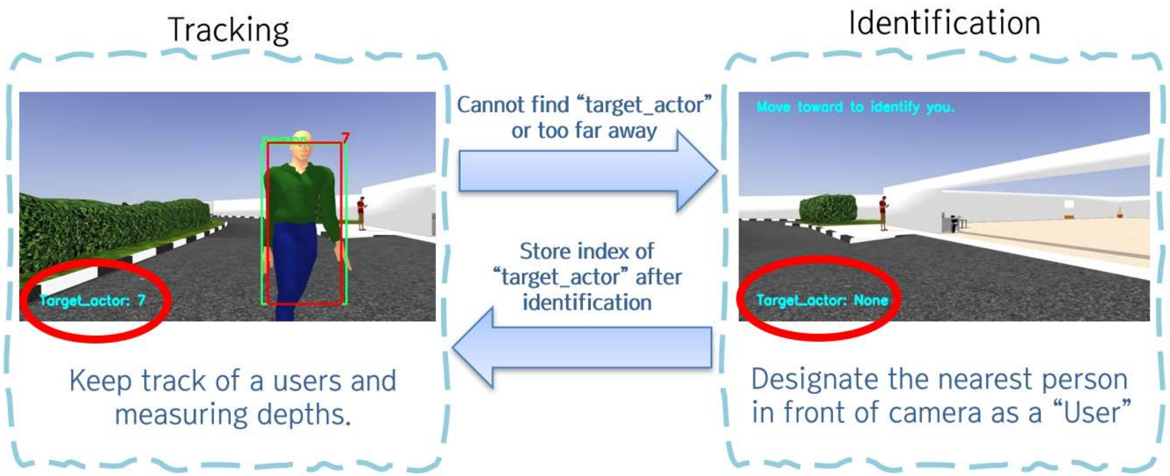
Literally just keep track of a user… Do not know if he's actually a user.

cf) face recognition

There's a chance of misidentification during (Re)Identification

ex) Someone stand in front of a user in identification process.

Recap)



5 Limitation

4) Development Environment

Simulation Computer Spec.

CPU: Intel i7-8700
Memory: 32GB
GPU: Nvidia Titan Xp

Server Computer Spec.

AWS EC2 T2

Not Enough spec. for Running DL Model

5 Limitation

5) Saving Information in App.

A structure that does not store information in the front-end,
but continuously receives information through the back-end

→ **Inefficient structure using information in App**

Use context or redux →**Global variables can be used within the front project**

6

Expectation

6 Expectation

Expectations and Effects

1 Accurate Performance

if there were enough resources to implement better hardware or robots, more accurate performance could be expected.

2 Apply Cloud

If you make a robot based on the cloud and implement additional functions, it can be used in various places

3 Project scalability

Applicability of scalability of robot workforce to care work

User-friendly features can be implemented

through additional app feature improvements

THANK YOU