ChaseMe Alarm Clock

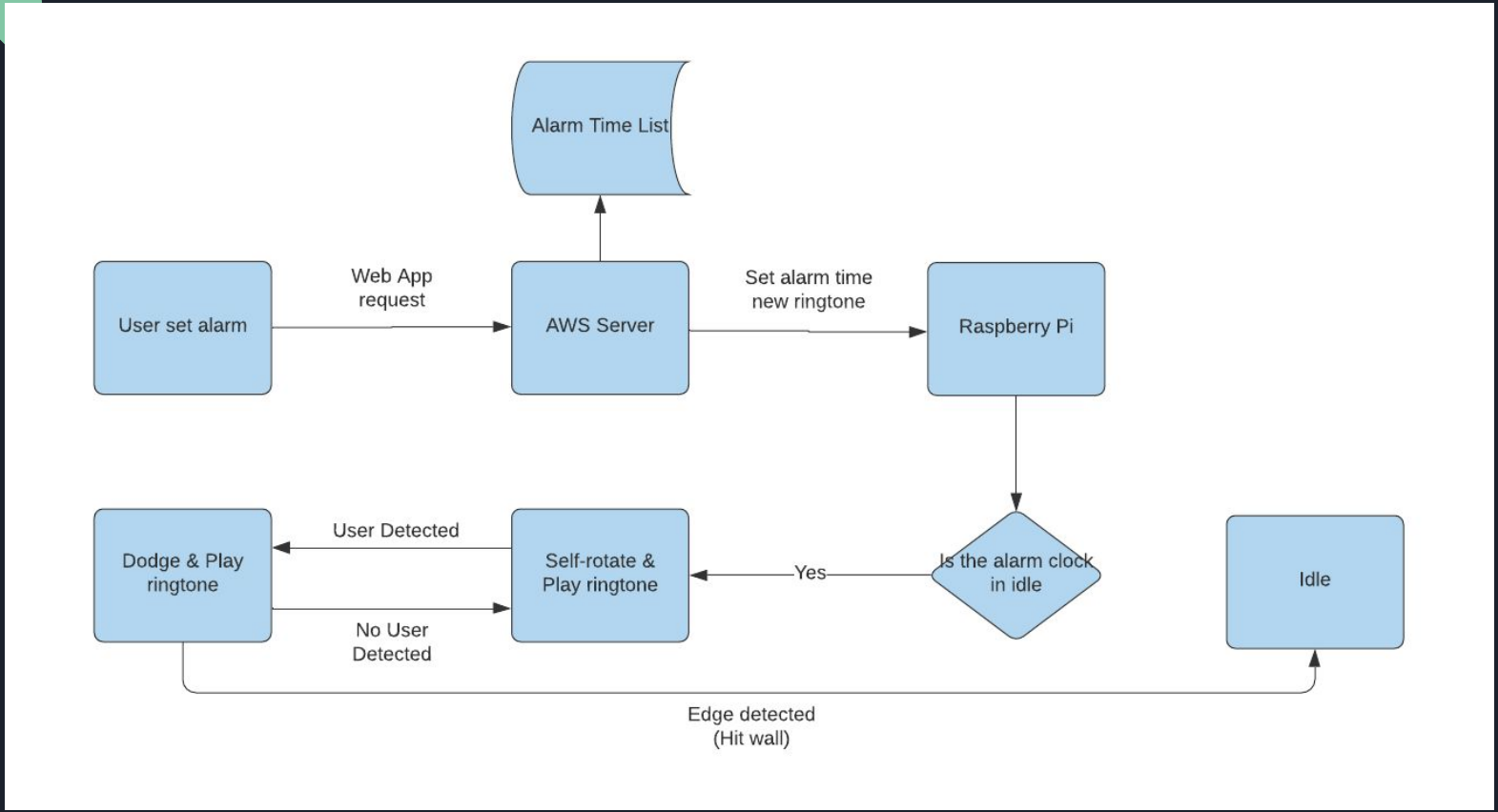
Team B1: Peizhi Yu, Yuhan Xiao, Echo Gao

Use Case

Have you ever had difficulties waking up in the morning? Have you ever turned off your alarm on your phone and keep napping?

We designed a smart alarm clock robot that can actually wake you up! It will run away from the person when he or she tries to catch it and turn it off.

Flow Diagram



General Requirements

- Only 1 person involved
  - The alarm clock will recognize the approaching person and move away from that person to avoid being caught
  - having 2 people in video capture at the same time will result in undefined behavior
- Flat surface
  - Most common situation for an indoor environment
- Unknown number of obstacles
  - User's personal items' placement is usually unknown
  - The alarm clock will try to avoid all obstacles
- Bright environment (with light on)
  - Chasing involved, user needs a bright environment to locate the alarm clock

Requirements (user behavior)

- User can approach from any direction, chase happens when user is detected
- User Route
  - The user can move freely relative to the bot
  - The user should move at a reasonable “walking speed”
- The robot should be able to track the user at 85% success rate
  - A successful dodge means the bot is able to lock its camera on the user throughout the chase
  - Previous “Following Robots” costs thousands of dollars, dozens of camera installed, and still lose track of the user
  - (Gita) https://www.youtube.com/watch?v=3fH_wku4FXM
- A chase stops when the robot is forced into an edge or a corner of the room by the user

Detailed Requirements (hardware)

- The alarm clock should be self-rotating at appropriate speed
  - Responsive enough to upcoming user while
  - We will sweep the rotation speed from 1 rpm to 0 rpm to determine the highest speed at which our CV algorithm can accurately detect the user
- The alarm clock should activate when user is detected
- The alarm clock should stop moving when user is not detected
- The userActivation delay, from user being in-range to robot starting to move, is under 0.25s
  - Larger delay may allow the user getting to the bot without chasing
  - Larger delay may also reduce user experience

Detailed Requirements (Computer Vision)

- Frames per second
  - Achieve 5 fps
  - Usually takes around more than 1 second to move from the centre of the camera out of the lens
- Human recognition rate at 90%
  - With 5 fps, even with an unrecognized frame, the robot will likely catch up
- False positive under 10%
  - A higher value might affect user experience
  - The robot, by design, won't react to non-moving object. So even the robot recognize something else as the user, it won't react accordingly

Detailed Requirements (Software/Web App)

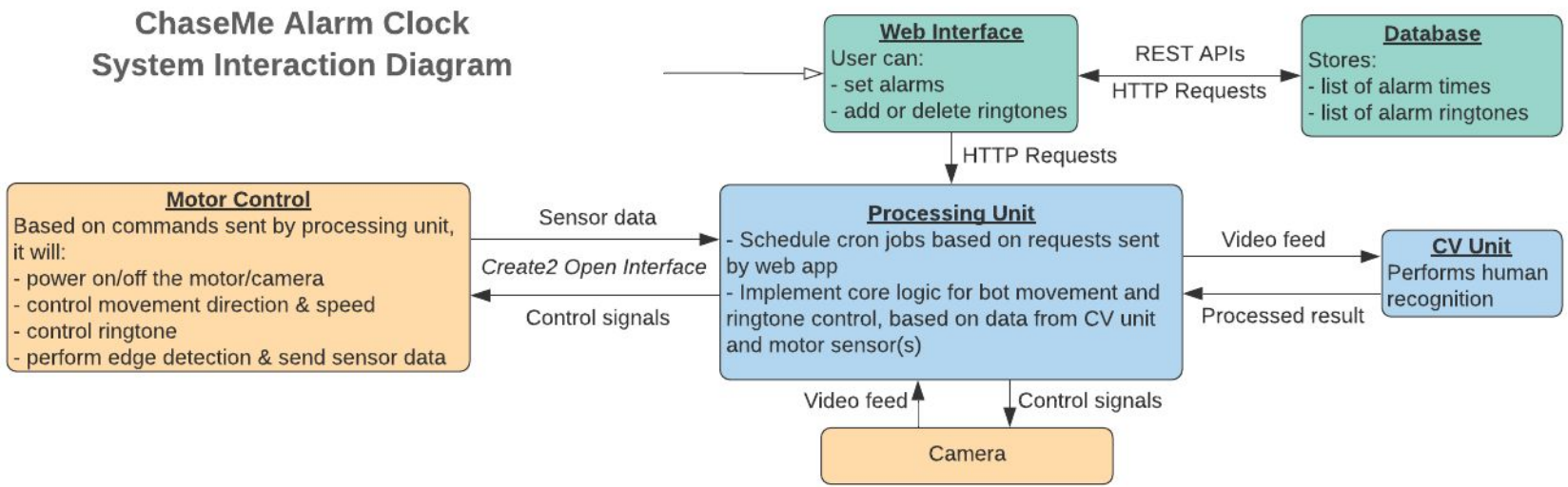
- Ability to reliably start the bot at a preset time in the future
  - ring time > 1s away from set time (latency)
  - ring time < 24 hours away from set time
- Communication latency between web app and the bot < 1s, including when:
  - starting the chase, from web app to bot
- Alarm is ad-hoc, instead of periodic
- But can set multiple alarms queued up for future

Solution Approach (hardware setup)

- Create2 robot base that supports edge detection
- A processing unit (Raspberry Pi or Nano Jetson 2) for controlling the base and running CV algorithm
- A depth camera like the Realsense for video capture and user distance measurements
- A webapp that allows the user to customize ringtone and control the robot

The processing unit and the camera will be attached to the robot base

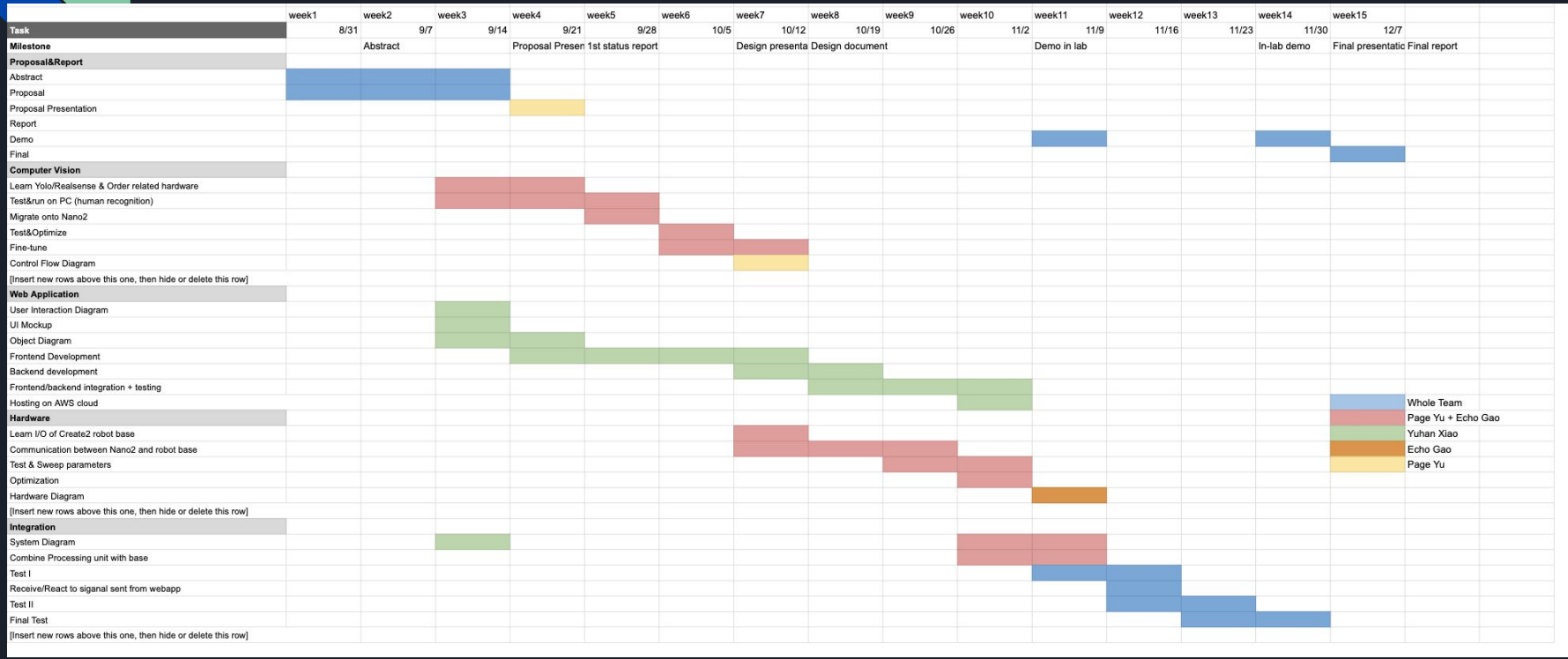
System Diagram



Testing

- Only 1 person involved
  - Test with multiple people appearing, the alarm clock should halt its action
- Unknown number of obstacles
  - Test with different numbers of obstacles that’s changing positions.
- Alarm clock activation
  - Test the robot with the user at different distance
  - Check if the bot can recognize the person in a single rotation
- Activation delay
  - Test if the activation delay is noticeable to the user, and if the user can walk up to the bot without chasing
- Communication latency between web app and the bot
  - Time the delay from the set alarm time to the bot’s LED light up/alarm turned on

Schedule & tasks & division of labors



	week1	week2	week3	week4	week5	week6	week7	week8	week9	week10	week11	week12	week13	week14	week15	
Task	8/31	9/7	9/14	9/21	9/28	10/5	10/12	10/19	10/26	11/2	11/9	11/16	11/23	11/30	12/7	
Milestone		Abstract		Proposal Presen	1st status report		Design presenta	Design document			Demo in lab			In-lab demo	Final presentatic	Final report
Proposal&Report																
Abstract																
Proposal																
Proposal Presentation																
Report																
Demo																
Final																
Computer Vision																
Learn Yolo/Realsense & Order related hardware																
Test&run on PC (human recognition)																
Migrate onto Nano2																
Test&Optimize																
Fine-tune																
Control Flow Diagram																
[Insert new rows above this one, then hide or delete this row]																
Web Application																
User Interaction Diagram																
UI Mockup																
Object Diagram																
Frontend Development																
Backend development																
Frontend/backend integration + testing																
Hosting on AWS cloud																Whole Team
Hardware																Page Yu + Echo Gao
Learn I/O of Create2 robot base																Yuhan Xiao
Communication between Nano2 and robot base																Echo Gao
Test & Sweep parameters																Page Yu
Optimization																
Hardware Diagram																
[Insert new rows above this one, then hide or delete this row]																
Integration																
System Diagram																
Combine Processing unit with base																
Test I																
Receive/React to siganal sent from webapp																
Test II																
Final Test																
[Insert new rows above this one, then hide or delete this row]																