Smart Public Restroom

**Phase 3: Development Part 1**

Planning and Design:

* Consult with disability advocates, caretakers, and occupational therapists to understand key accessibility needs and challenges.
* Research best practices for accessible and universal design of restrooms.
* Map out restroom layout with rough blueprints and 3D models. Plan placement of fixtures, doors, signage, etc.
* Design electronic and automated components like doors, faucets, soap dispensers, hand dryers.

Prototyping:

* Create rough prototypes of key components like doors, sinks, signs.
* Test prototypes with focus groups and revise designs based on feedback.
* Develop Raspberry Pi proof-of-concept with basic functionality.
* Refine electronics, sensors, audio prompts iteratively through testing.

Development:

* Code software and program Raspberry Pi functionality using Python.
* Integrate components like automated doors, touchless fixtures, lighting.
* Install audio prompts, occupancy alerts, and accessibility signage.
* Make final refinements to maximize universal access.

Deployment:

* Identify initial sites for installation, focusing on high-traffic public spaces.
* Install restroom units fully, potentially keeping traditional restrooms also in place during transition.
* Provide training to maintenance staff to properly operate and repair units.
* Gather user feedback post-deployment to improve designs and experiences.

# Raspberry Pi Control Code for Smart Public Restroom:

# Import required libraries

import RPi.GPIO as GPIO

import time

import pygame

# Set up GPIO pins

doorLock = 4

doorSensor = 17

sinkSensor = 27

lightSensor = 22

# Set pin modes and initial states

GPIO.setmode(GPIO.BCM)

GPIO.setup(doorLock, GPIO.OUT, initial=GPIO.HIGH)

GPIO.setup(doorSensor, GPIO.IN)

GPIO.setup(sinkSensor, GPIO.IN)

GPIO.setup(lightSensor, GPIO.IN)

# Initialize pygame for audio

pygame.init()

occupy\_chime = pygame.mixer.Sound("./occupychime.wav")

vacant\_chime = pygame.mixer.Sound("./vacantchime.wav")

# Main loop

while True:

# Check door sensor

if GPIO.input(doorSensor):

# Unlock door for 5 secs

GPIO.output(doorLock, GPIO.LOW)

time.sleep(5)

GPIO.output(doorLock, GPIO.HIGH)

# Check sink sensor

if GPIO.input(sinkSensor):

# Trigger sink automation

print("Sink used")

# Check light sensor

if GPIO.input(lightSensor):

# Turn on lights

print("Lights on")

else:

print("Lights off")

# Play audio occupancy alerts

if <stall is occupied>:

occupy\_chime.play()

if <stall is vacant>:

vacant\_chime.play()

time.sleep(0.5)

*This uses the GPIO pins and pygame library to control doors, fixtures, lights, and audio alerts. The main loop runs continuously checking sensors and triggering actions. Additional code would be needed to integrate other components and features per the project requirements.*