Escape Mission: SOS Morse Code

Pennsylvania State University, Abington

Spring 2022- 4/29/22

CMPSC 488/ IST 440W

Team 3

Professor: Joseph Oakes

Project Manager: Victoria Douglas

Table of Contents

Table Of Figures

1.0 Mission Statement

We will be making a communication system utilizing Morse code with escape room elements. The purpose of the application is to establish and conduct communication with mission control back on Earth to coordinate a rescue.

The application will allow users to input messages in English that will be broadcast into space. These messages will be translated to Morse code in the application then sent to the M5 stack to be broadcast. The Morse code messages will be broadcast into space via the LED and Buzzer Module through the M5 stack using Morse code conventions. The M5 stack will also be able to receive incoming morse code broadcasts via photoresistor and microphone sensors. Received Morse code messages will be sent back to application and translated into English for the user.

To use the communicator, the user will have to complete a puzzle where a word will be displayed in morse code, and they must translate it (with the use of a guide sheet) and choose the correct word from the options provided.

1.1 Problem Definition

Being stranded on Moon Base Alpha there is little hope of rescue without establishing communication with mission control back on Earth. To establish communication, mission control must first know where to look for our messages. Due to the sheer size of the solar system, we must ensure that communication is reliable and efficient over such long distances. The astronauts will find themselves in a dire situation if their communication system cannot reach anyone. Their job is to send messages to the rescue team. The maintenance worker is responsible for the system functioning correctly, while the administrator will oversee the messages between the astronauts and the rescue team. We will have a better chance of being rescued if we establish contact with mission control on Earth.

1.2 Project Stakeholders

Name	Title	Role	Contact Information
Matthew Coutts	Team Leader	Admin	Mec6250@psu.edu
Almazbek Akhunbaev	Team Member	Astronaut	ava6100@psu.edu
Alexander Djordjevic	Team Member	Astronaut	Aqd5694@psu.edu
Md Mollah	Team Member	Astronaut	Mmm7382@psu.edu
Bradley Whitebread	Team Member	Maintenance	bew5294@psu.edu
Joseph Oakes	Professor	Professor	jxo19@psu.edu
Victoria Douglas	Project Manager	Project Manager	vgd5033@psu.edu

2.0 User Stories

2.1 - As an astronaut, I want to translate morse code so that I can understand the message faster.

2.2 - As an astronaut, I want to be able to send a message that will repeat, so that I can focus on other work.

2.3 - As an astronaut, I want to send messages so that I can communicate with the astronauts.

2.3 - As an astronaut, I want a log of messages so that I can read them later.

2.4 - As an astronaut, I want a log of messages so that I can refer to them.

2.5 - As an administrator, I want to be able to remove old messages so that we can prioritize important messages.

2.6 - As an administrator, I want to be able to see if messages are delivered successfully so we can ensure correct functionality.

2.7 - As a maintenance worker, I want to be able to test the morse code translator, so we know the messages we send are correct.

2.8 - As an astronaut I want a way to signal to my location so that mission control will know where to look for us.

3.0 Use Cases

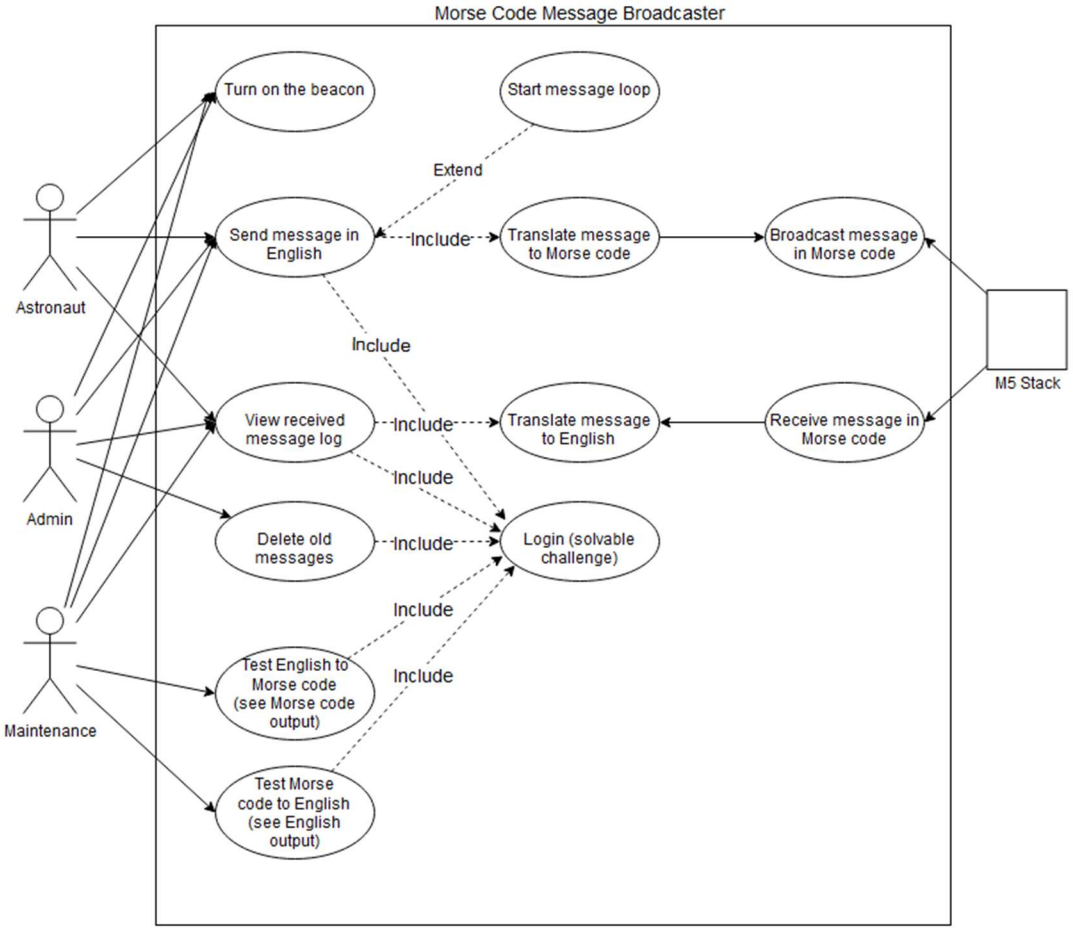


Figure 1 Use Case Diagram (Dwivedi, Use case diagram - Software Design: Modeling with UML) (Dwivedi, Use-cases - Software Design: Developing Effective Requirements)

4.0 Functional Requirements

4.1 - The system will translate inputted English messages into Morse code.

4.2 - The system will translate received Morse code messages into English.

4.3 - The application will contain a message log that will be accessible by users.

4.4 - The system will output a random word or phrase in morse code that must be matched to log in.

4.4.1 - The system will vary the number of options depending on the user level. (3 for the basic user, 5 for maintenance user, 7 for admin user)

4.4.2 - The system shall allow the user two tries before the Morse code phrase updates.

4.5 - The application will allow a user with admin privileges to delete messages.

4.6 - The application will display what morse code would be broadcast for a given English text input.

4.7 - The application will display what English text would be translated for a given Morse code input.

4.8 - The system will have an emergency button which once pressed will allow users to send one message.

4.9 - The system will have an emergency button which once pressed will allow users to only view the past five messages.

5.0 Technical Requirements

5.1 - M5 Stack Controller

5.1.1 - The M5 Stack controller will run software that produces morse code through the user pressing a button or typing in text. This will be translated and outputted to a data pin, this data pin will connect to the LED.

5.1.2 - The M5 Stack will also provide power to the LED via the 5v or 3.3v port and GND, a resistor may be used here.

5.1.3 - While this is occurring, the M5 Stack will simultaneously push the translated message to an application on our Control Operations Center (Raspberry Pi) via Wi-Fi.

5.2 - LED (Light Emitting Diode) Strip with addressable port (ALTILOVE LED STRIP) or (MATRIX KIT)

5.2.1 - The LED is connected to the M5 Stack

5.2.2 - The LEDs will flash in intervals to display a morse code message

5.3 - Resistors (tentative since LEDS take 5V)

5.3.1 - Used on the ground wire to the LED

5.3.2 - Resistors prevent LEDs from burning out quickly

5.3.3 - We will use Ohms law to determine correct resistors for certain components

5.4 - Multicolor wires (male to male & male to female)

5.4.1 - Used to connect various electrical components

5.4.2 - Will color code wires

5.4.2.1 - RED – positive terminals (5V+)

5.4.2.2 - BLACK – negative terminals (Ground)

5.4.2.3 - OTHER COLOR – Signal wires for digital (PWM) and analog controls

5.5 - Software Interface (GUI Control Operations Center)

5.5.1 - Translate morse code into English that it receives from the M5 Stack via Wi-Fi

5.5.2 – User-friendly design with a message log

5.5.3 - Control Operations Center will be hosted on a Raspberry Pi

5.6 - Breadboard (Prototype wiring)

5.6.1 - We will use the breadboard to wire all the components together

5.6.1.1 - M5 Stack will provide 5V to our + and GND to -

5.6.1.2 - Resistors will be used between + and LEDs

5.6.1.3 - We can also include a switch here between the LED and M5 Stack

5.6.1.4 - If needed to add any other components the breadboard is interchangeable

6.0 Data Flow Diagram

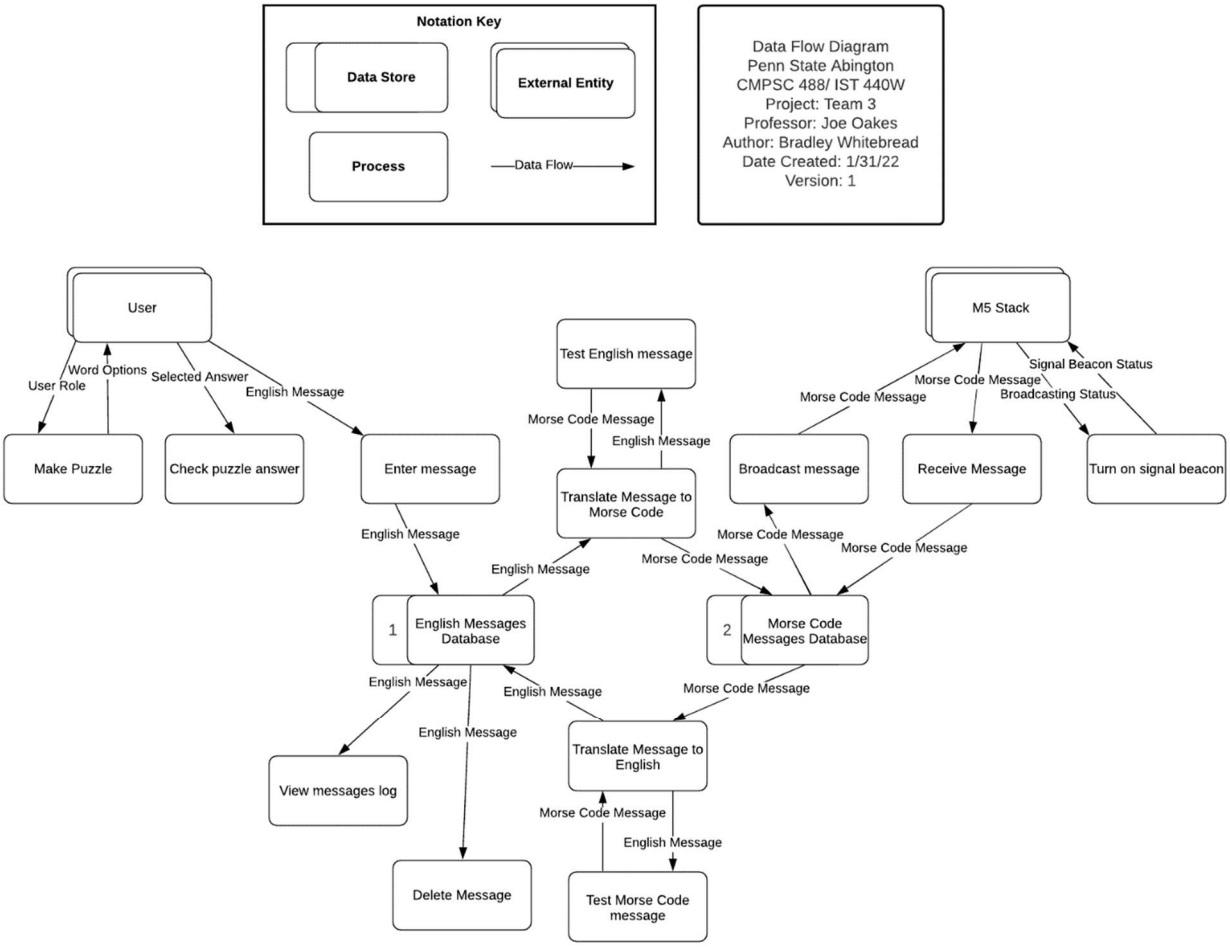


Figure 1 Data Flow Diagram (Wick)

7.0 Enterprise Relationship Diagram

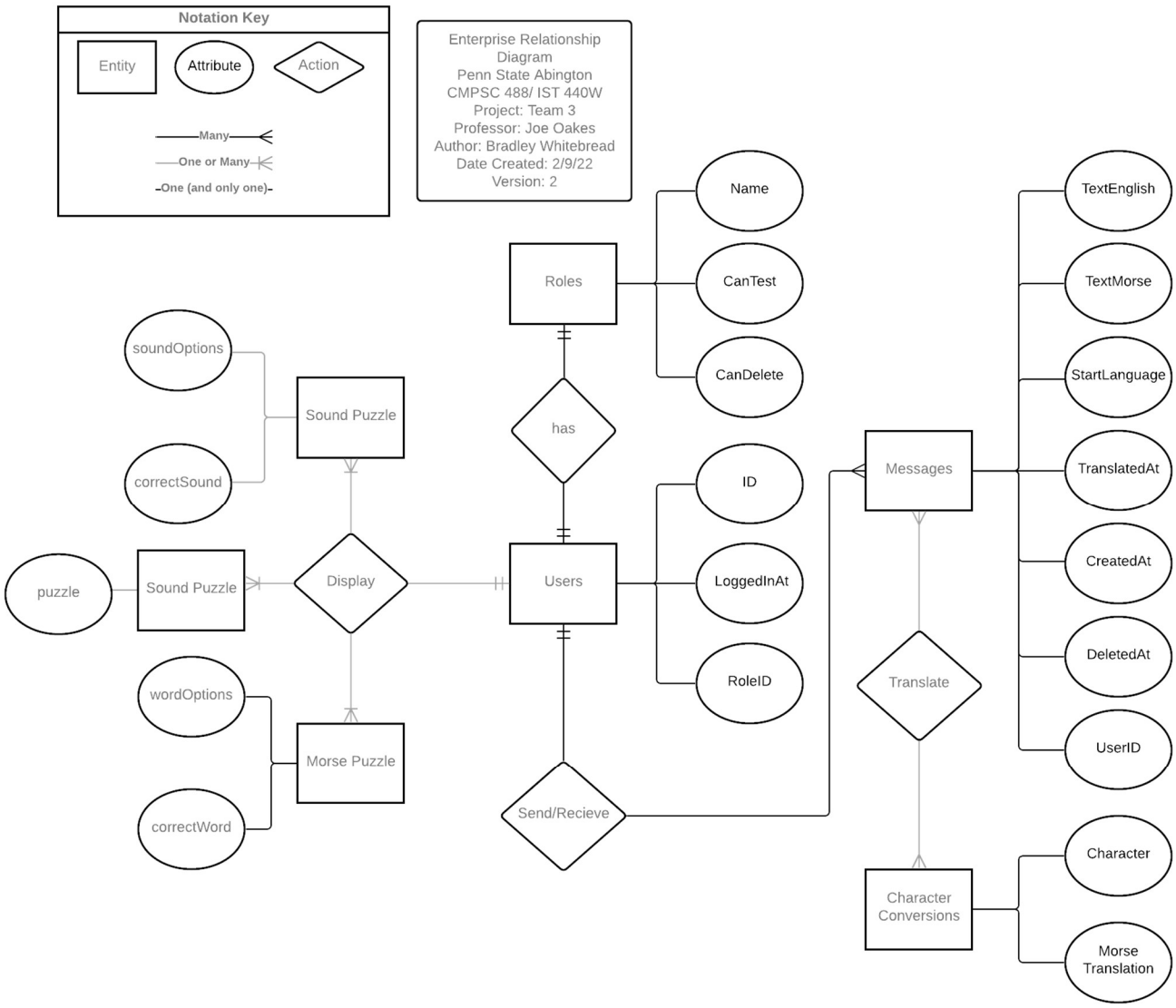


Figure 2 Enterprise Relationship Diagram

8.0 Class Diagram (UML)

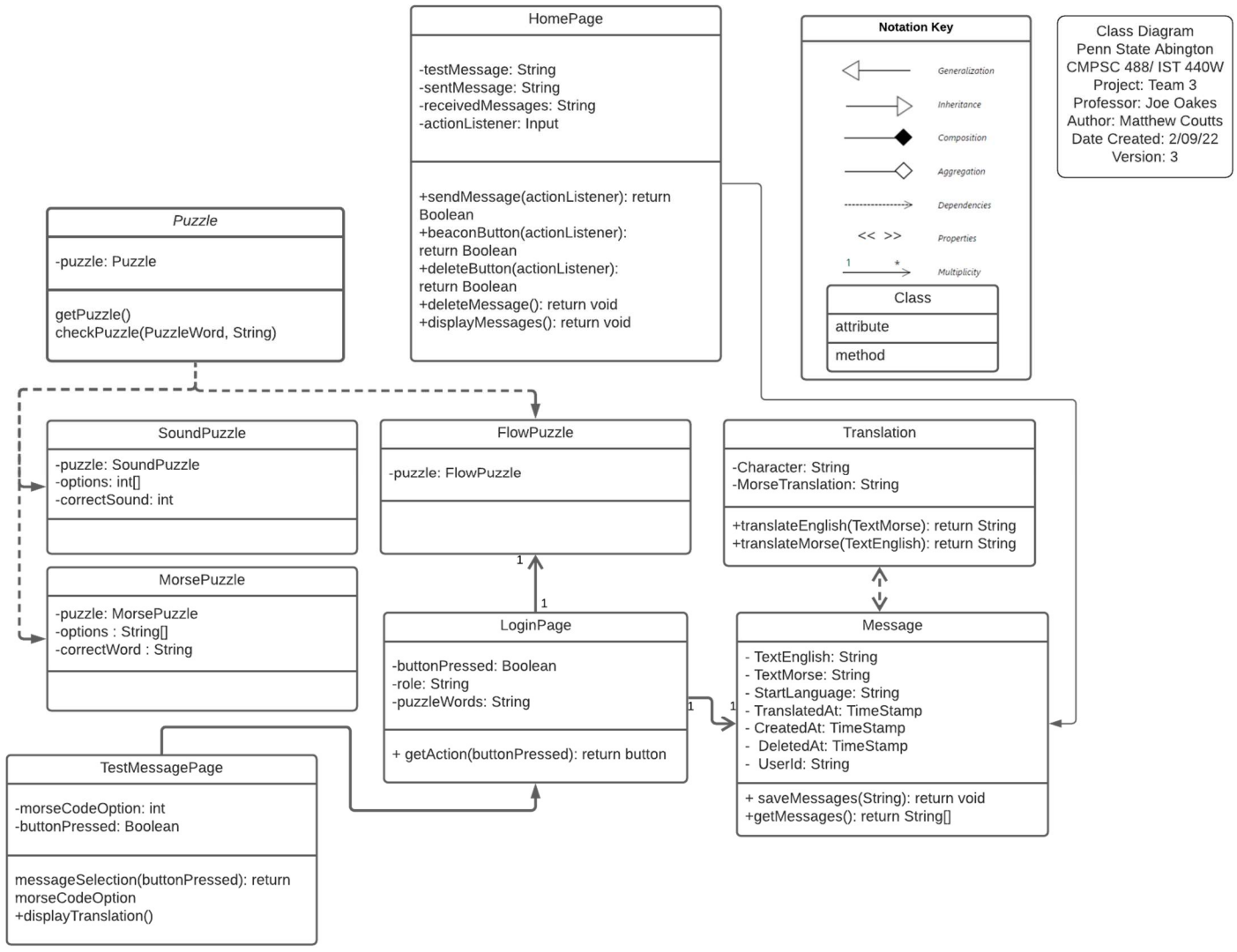


Figure 3 Class Diagram

9.0 Sequence Diagram (UML)

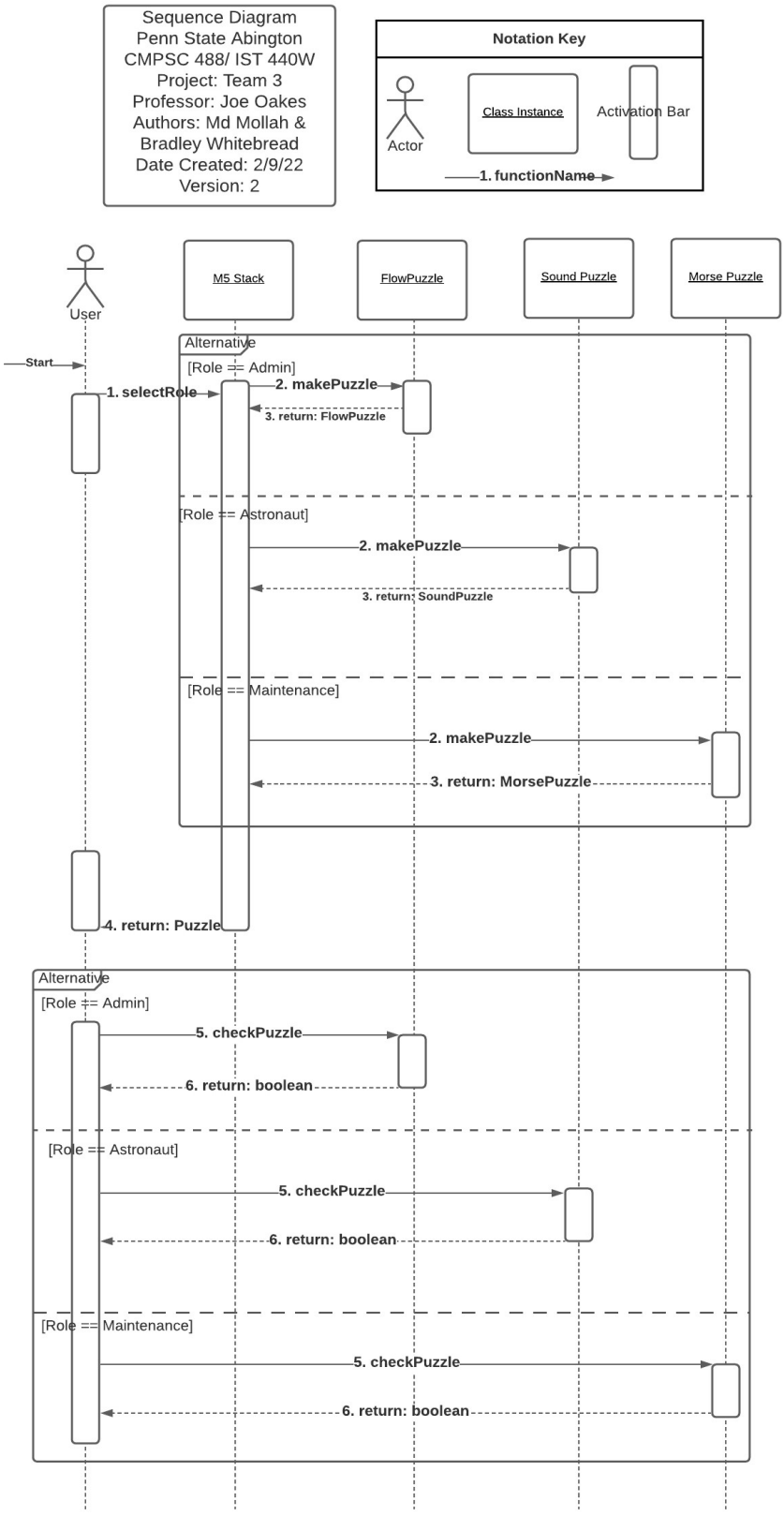


Figure 4 Login Sequence Diagram

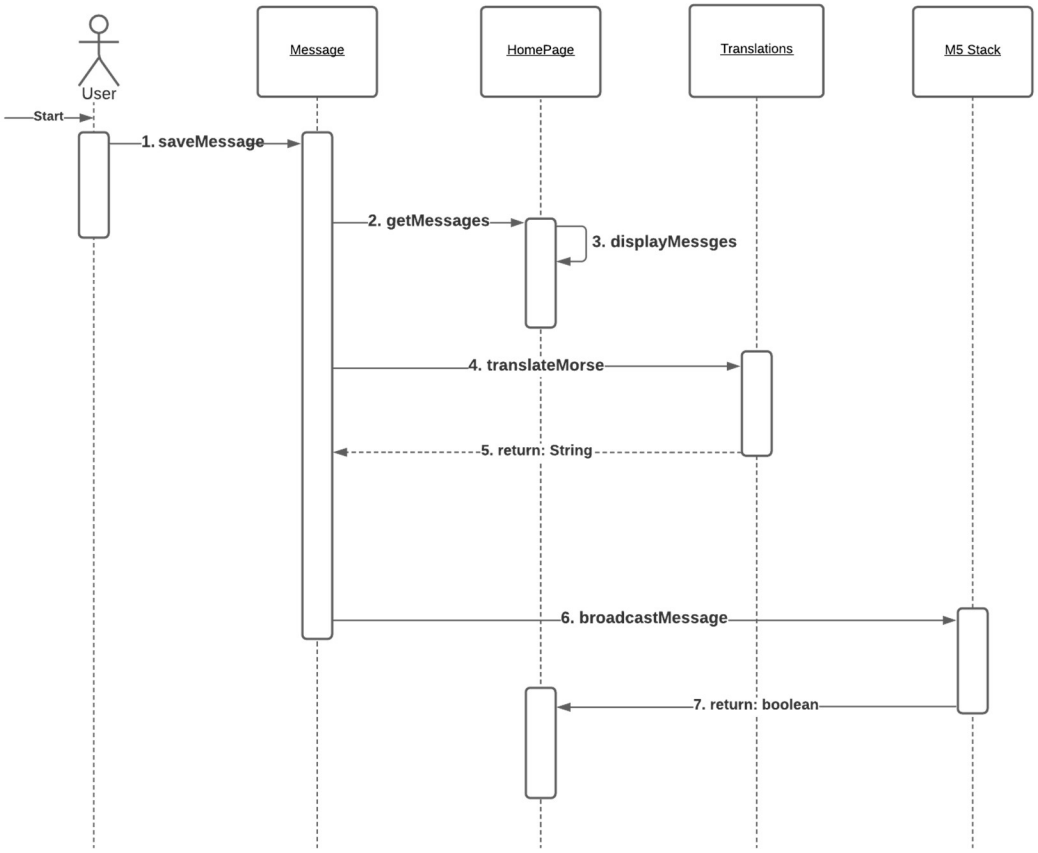


Figure 5 Broadcast Message Sequence Diagram

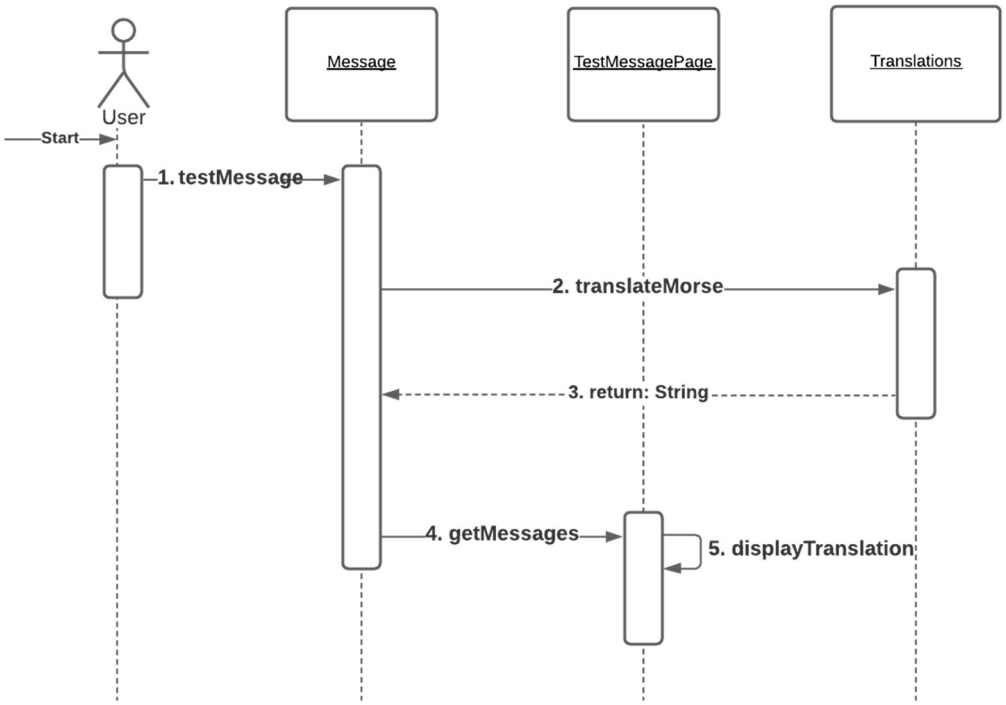


Figure 6 Test Messages Sequence Diagram

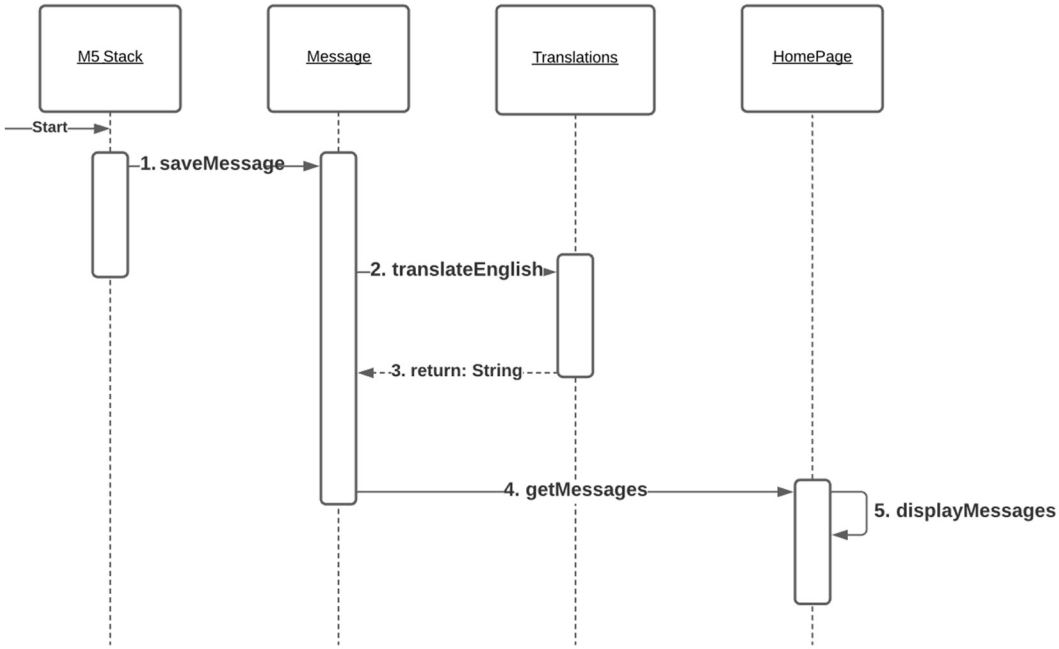


Figure 7 Receive Messages Sequence Diagram

10.0 Activity State Diagram (UML)

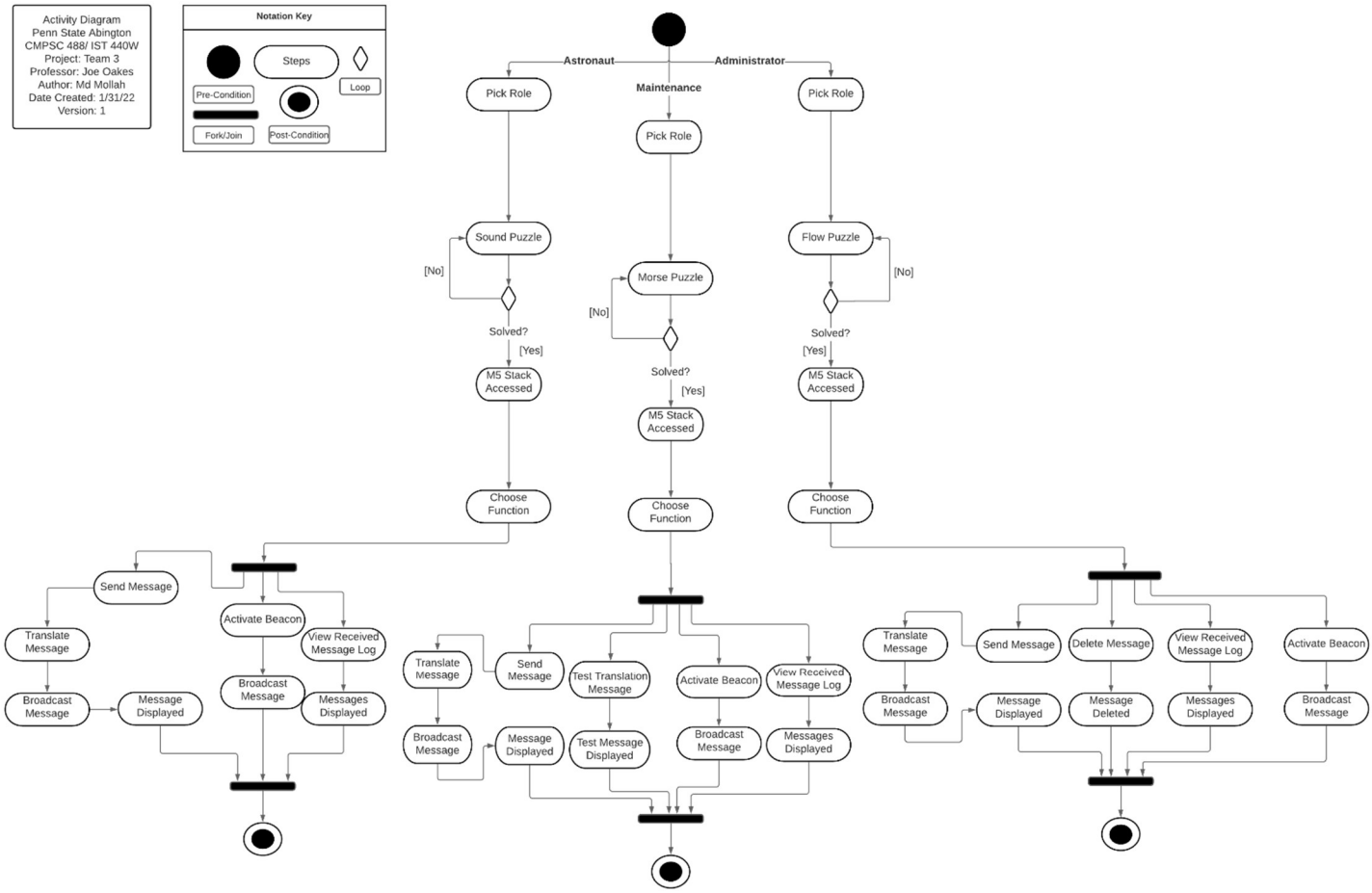


Figure 8 Activity Diagram

11.0 Paper Prototypes

Paper Prototypes
Penn State Abington
CMPSC 488/ IST 440W
Project: Team 3
Professor: Joe Oakes
Author: Almazbek Akhunbaev
Date Created: 1/31/22
Version: 4

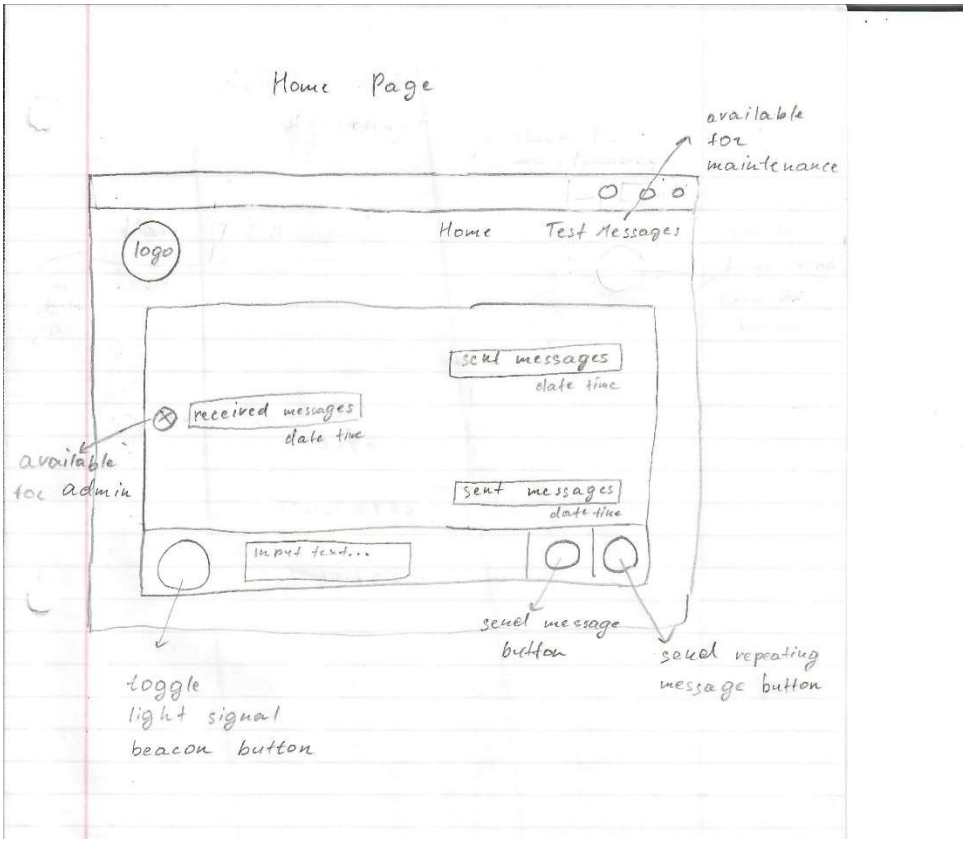


Figure 9 Home Page Paper Prototype

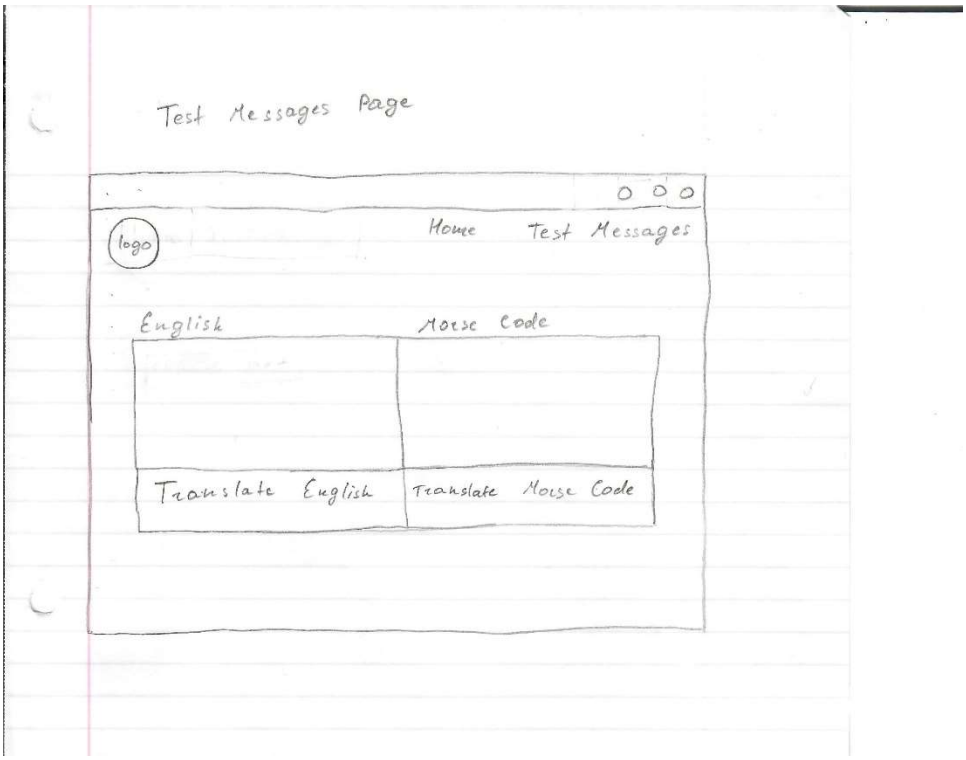


Figure 10 Test Messages Page Paper Prototype

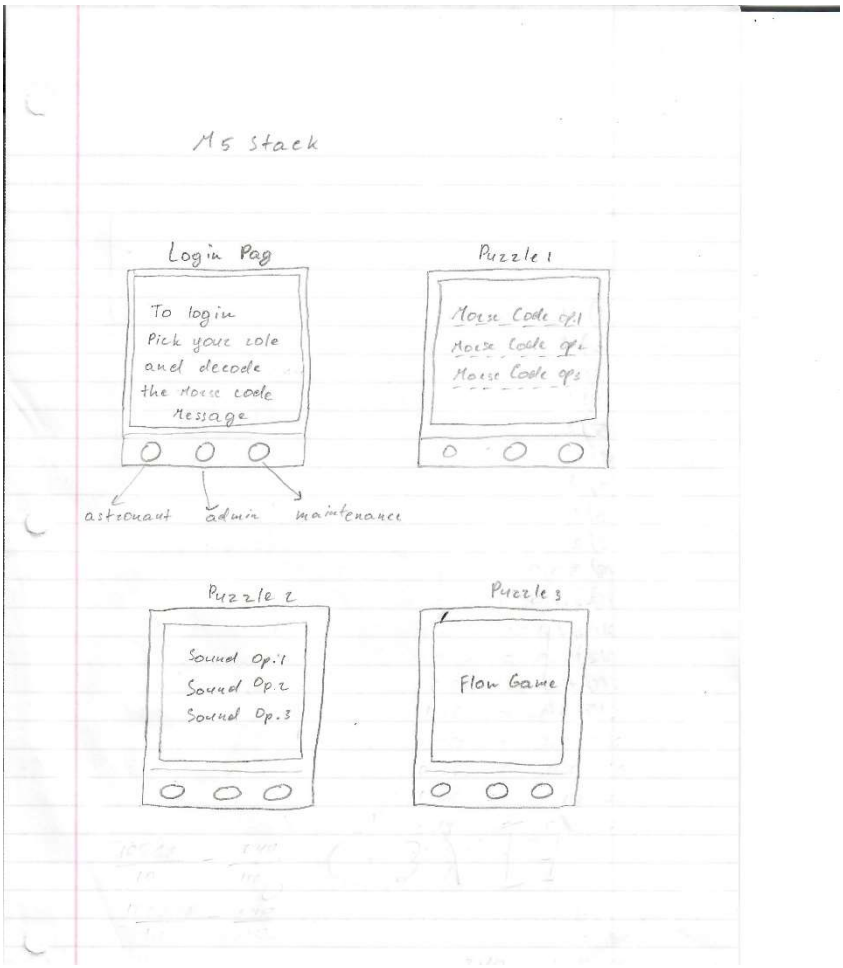


Figure 11 M5Stack Prototype

12.0 Wireframes Design

Wireframe Design
Penn State Abington
CMPSC 488/ IST 440W
Project: Team 3
Professor: Joe Oakes
Author: Almazbek Akhunbaev
Date Created: 1/31/22
Version: 3

Figure 12 Home Page

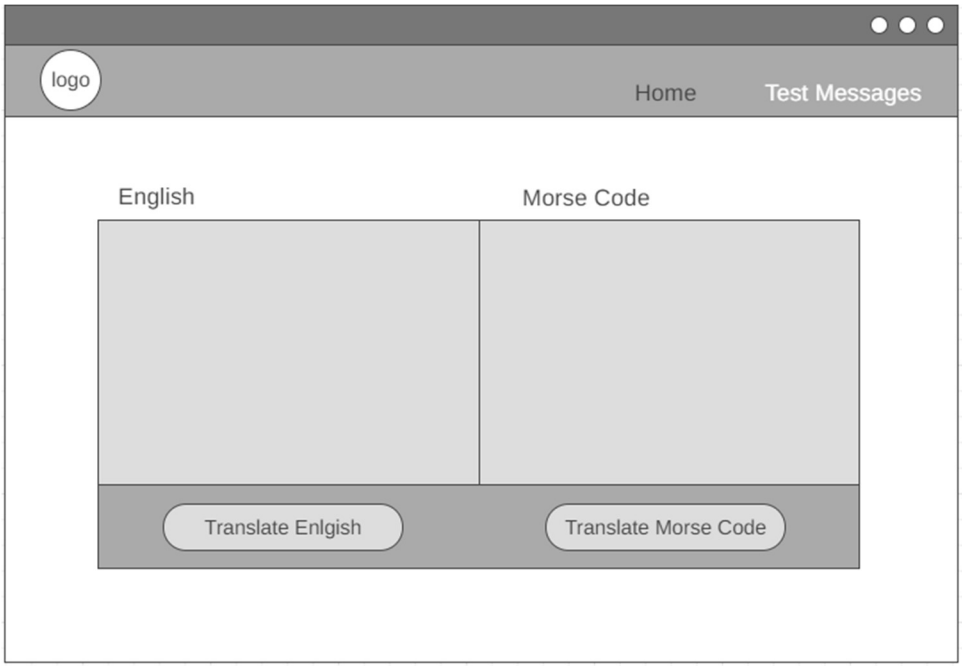


Figure 13 Test Messages Page

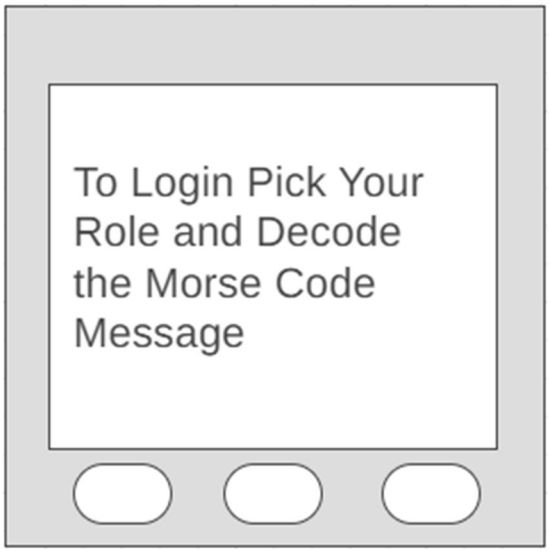


Figure 14 M5Stack Login Page

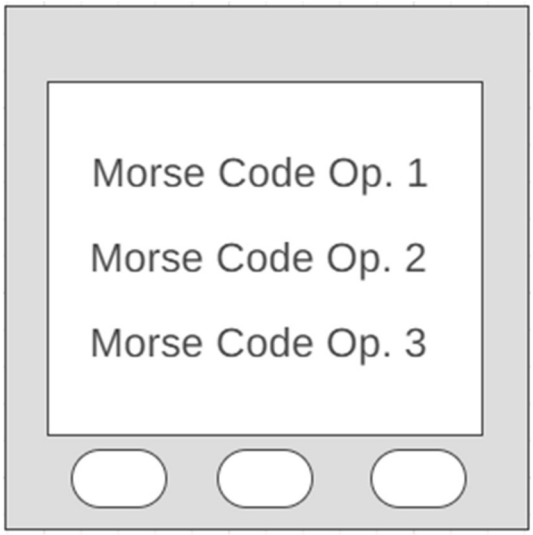


Figure 15 M5Stack Puzzle 1

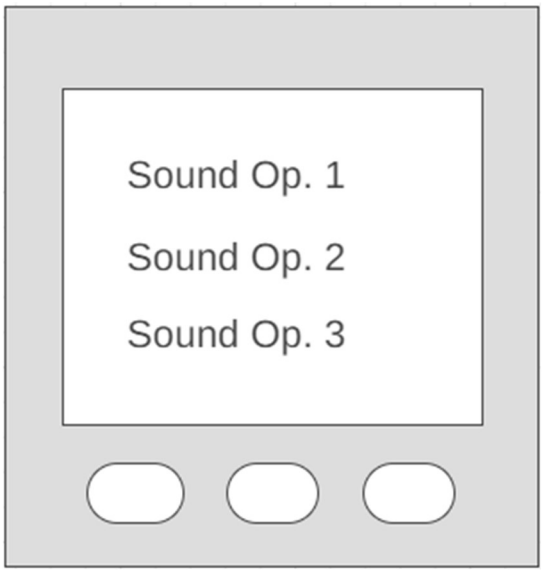


Figure 16 M5Stack Puzzle 2

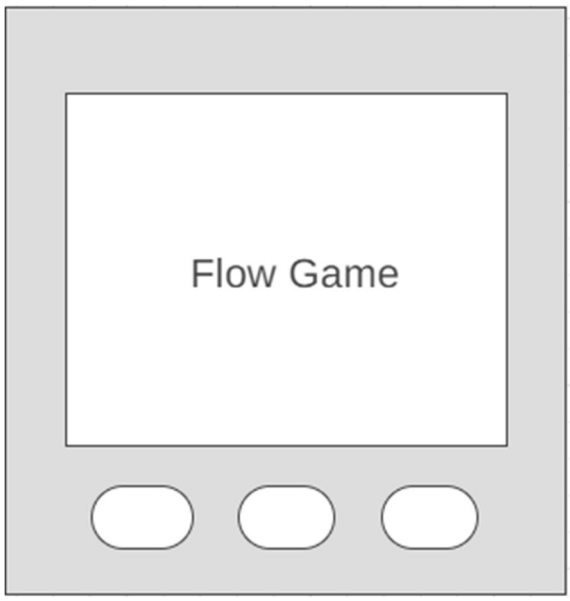


Figure 17 M5Stack Puzzle 3

13.0 Style Guide

Style Guide
Penn State Abington
CMPSC 488/ IST 440W
Project: Team 3
Professor: Joe Oakes
Author: Matthew Coutts
Date Created: 2/6/22
Version: 2

1
COLOR SWATCH

— Base Colors

— Primary Colors

— Secondary Colors

2
TYPOGRAPHY

— Font weights

Aa Segoe UI Light

Aa Segoe UI SemiLight

Aa Mulli Regular

Aa Segoe UI SemiBold

Aa Mulli Bold

Aa Segoe UI Black

— Headings

H1-60px Font Weight- **Bold** Line Spacing- **72px** Character Spacing- **16**

H2-48px Font Weight- **Bold** Line Spacing- **61px** Character Spacing- **16**

H3-40px Font Weight- **Bold** Line Spacing- **58px** Character Spacing- **16**

H4-30px Font Weight- **Bold** Line Spacing- **44px** Character Spacing- **16**

H5-24px Font Weight- **Bold** Line Spacing- **38px** Character Spacing- **16**

H6-18px Font Weight- **Bold** Line Spacing- **29px** Character Spacing- **16**

— Paragraph

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur

— Pointers

- Lorem ipsum dolor sit amet, consectetur adipisicing elite
- Lorem ipsum dolor sit amet, consectetur adipisicing elite
- Lorem ipsum dolor sit amet, consectetur adipisicing elite
- Lorem ipsum dolor sit amet, consectetur adipisicing elite

3
BUTTONS

— Colored

4

Figure 18 Style Guide (Popat) ("Color Palette: #1A374D #406882 #6998AB #B1D0E0 - Color Hunt") ("Color Palette: #FFF89A #FFC900 #086E7D #1A5F7A - Color Hunt")

14.0 Mockups Design

Mockups Design
Penn State Abington
CMPSC 488/ IST 440W
Project: Team 3
Professor: Joe Oakes
Authors: Alexander Djordjevic and Bradley Whitebread
Date Created: 2/6/22
Version: 2

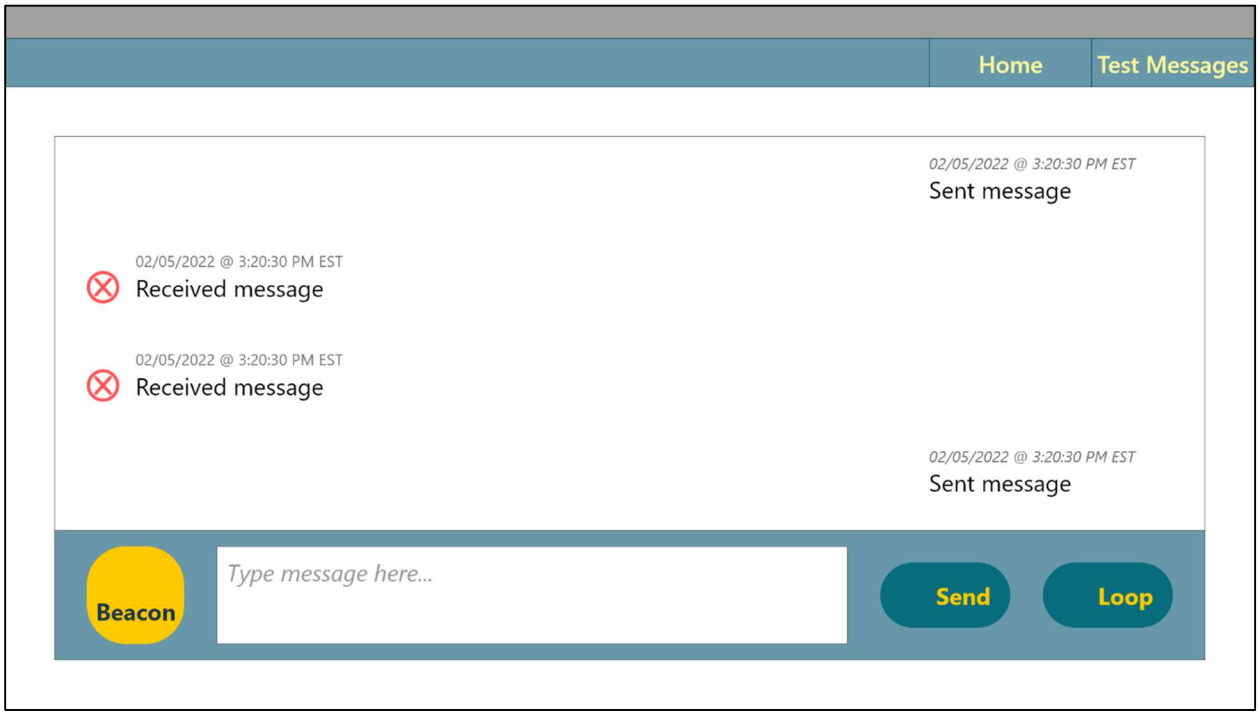


Figure 19 Home Page Mockup

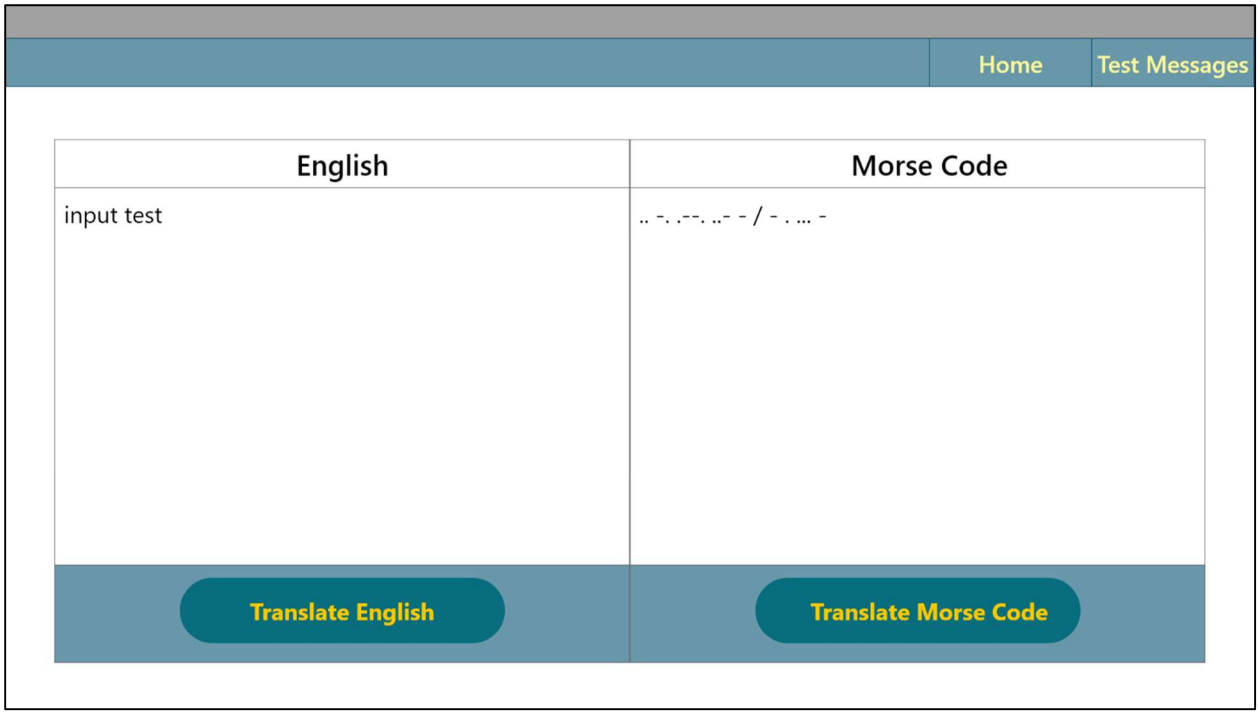


Figure 20 Test Page Mockup

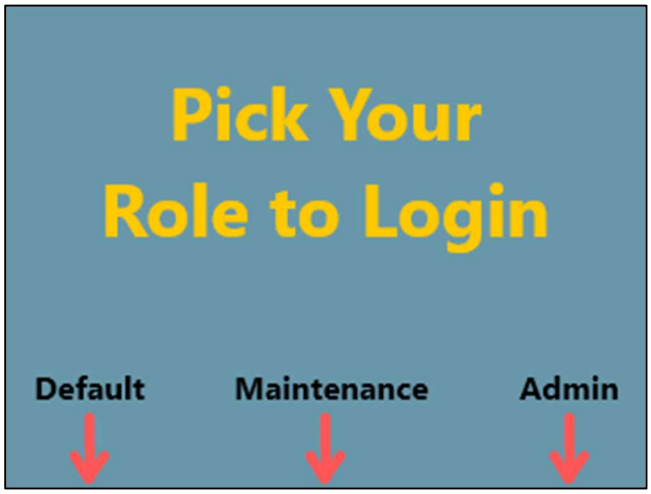


Figure 21 M5 Stack Role Selection Page

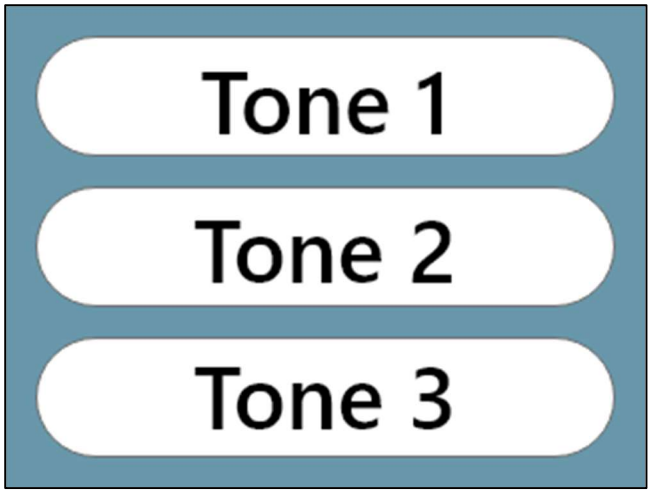


Figure 22 M5 Stack Default User Login Puzzle Page

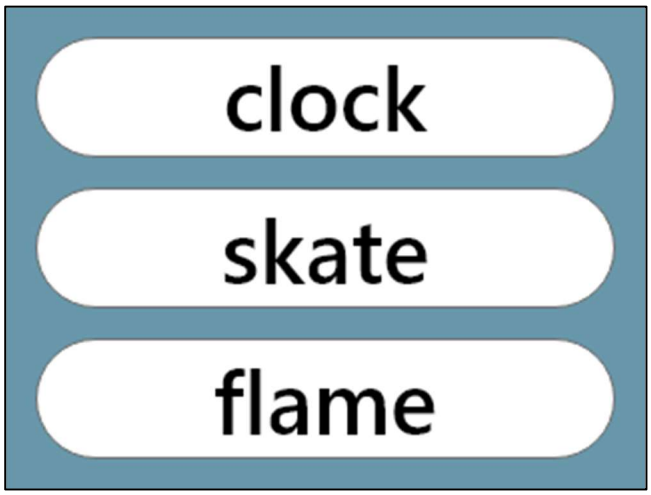


Figure 23 M5 Stack Maintenance User Login Puzzle Page

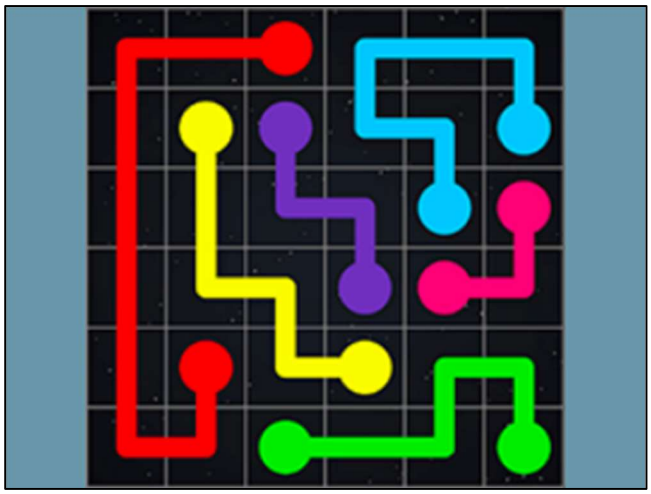


Figure 24 M5 Stack Admin User Login Puzzle Page (“Flow.”) (“Play Flow Lines.”)

15.0 Workflow Diagram

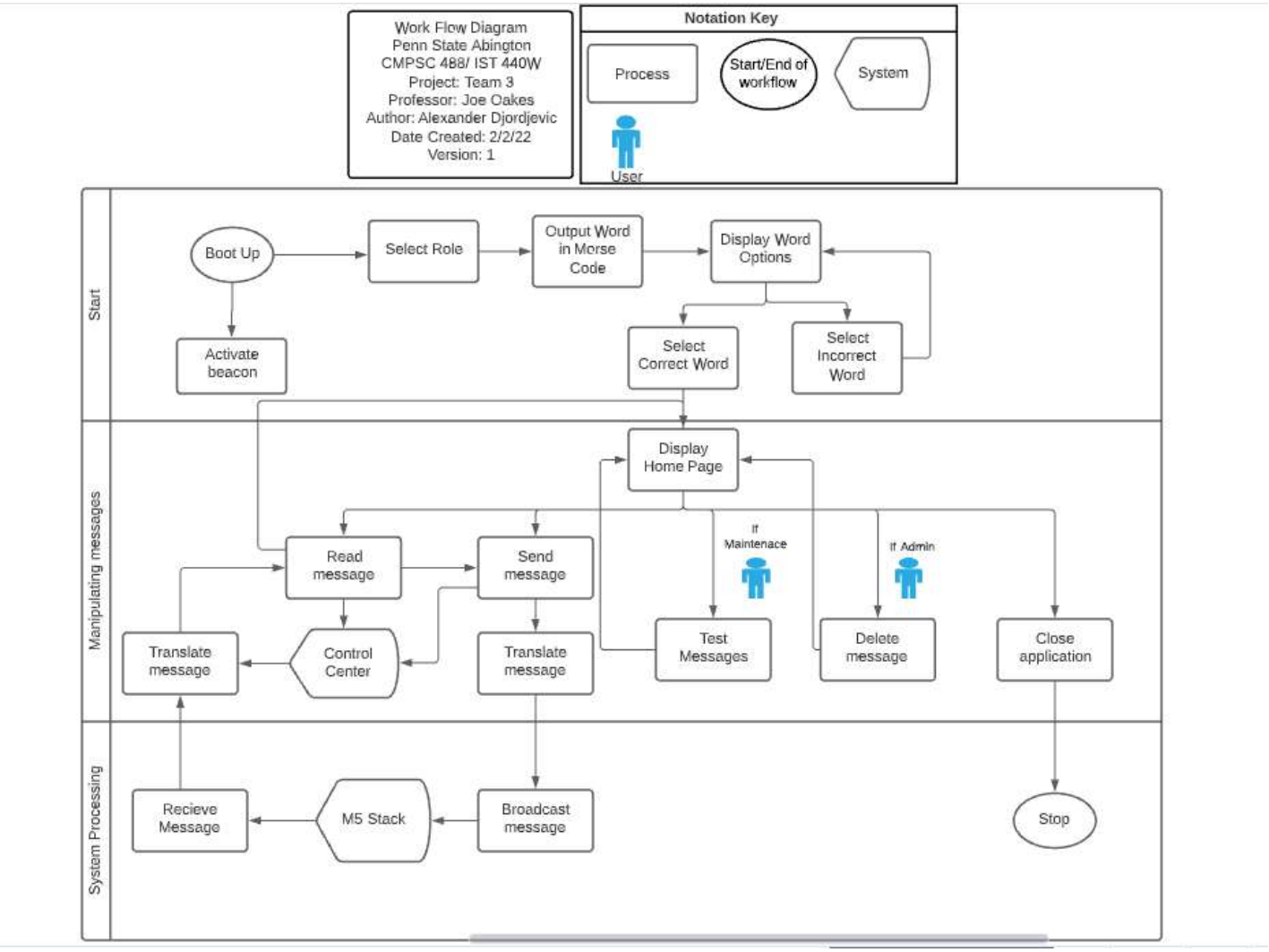


Figure 25 Workflow Diagram

16.0 Pair Programming

Teams	Team Member #1	Team Member #2
1	Bradley Whitebread	Md Mollah
2	Alexander Djordjevic	Almazbek Akhunbaev
1 + 2	Matthew Coutts	

16.1 Lessons Learned

16.1.1 – We were able to produce better quality code in pairs as opposed to solo programming.

16.1.2 – Errors were solved easier and faster.

16.1.3 – Ideas come from both developers which allowed us to think of different ways of approaching a problem.

16.1.4 – We shared knowledge to learn from each other.

16.1.5 – We were able to verbally program and talk while writing code helped us realize whether the approach was viable.

16.2 Team Tasks

16.2.1 - Team 1 Tasks

16.2.1.1 - Created a login page so users can choose their role and solve the puzzle assigned to that role.

16.2.1.2 - Created a flow puzzle that asks the user to connect dots to the specific colors until all colors are linked to their matching color. The flow puzzle is for users who choose maintenance as their login.

16.2.1.3 - Created a morse puzzle broadcasts where a five-letter word is broadcast in morse code the user must select the correct translation of the word in English from three options. Once the puzzle is successfully completed the user is logged in as an admin.

16.2.1.4 - Created a method that takes a message, translates it into morse code and broadcasts the message so the receiver can decode it.

16.2.2 - Team 2 Tasks

16.2.2.1 - GUI Application waits for the M5Stack Signal and opens the maintenance, admin, or default page based on the puzzle the user completed. The maintenance page allows to send message and test messages by translating them between English and Morse Code. The admin page allows to send messages and delete the sent messages. The default page only allows to send messages.

16.2.2.2 – Created a sound puzzle that plays three unique tones then one repeated tone. The user must decide which of the three tones was repeated. Once the puzzle is successfully completed the user is logged in as an astronaut.

17.0 Coding Conventions

17.1 Comments

Header comment should contain meta information regarding the document refer to example below.

Ex]:

"""

```
Project: Lab 1

Purpose Details: Client-Server capitalization application

Course: IST 411

Author: Joe Oakes

Date Developed: 1/19/2020

Last Date Changed:

Revision: 1

"""
```

Documentation of functions

Use `triple double quotes` explaining what a function does directly beneath the function signature

Ex]:

```
def doSomething():
    """returns the integer 5""" ← documentation comment
    Return 5
```

Multiline comments

For multiline comments use one '#' per line of the comment

Ex]:

```
# This is an example
# of a multiline comment
```

Spacing

Add a space after '#' character in comments

```
# Single line comment example
```

17.2 Whitespace

Indentation

Indentations should be 4 spaces long

Ex]:

```
class Example:
____def doSomething():
```

```
____ ____return 5
```

Line length

Each line should be no longer than 100 characters

Sequential empty lines

Two maximum sequential empty lines

Line endings (newlines)

Use '\n'

Block statements

`def test():` ← no space between closing parenthesis and colon in function definition

Control statements

`If a == b:` ← no space between end of control statements and colon

17.3 Naming

File Names

File names should be in UpperCamelCase AKA PascalCase

Ex]: ExampleFile.py

Folder Names

Folder names should be in PascalCase

Ex]: ExampleFolder

Class names

Class names should be in PascalCase

Ex]: class ClassName:

Function Names

Function names should be in lowerCamelCase

Ex]: def newFunction():

Variable Names

Variable names should be lowerCamelCase

Ex]: newVariable

Length

Keep names short the ideal length is around 8 characters. Names should be no more than 15 characters

Meaningful

Try to keep variable names meaningful

17.4 Error Handling

Error messages

Error messages should be formatted with the name of the error along with a brief description of the error.

Ex]: Error: Invalid symbol used

18.0 Work Breakdown Structure

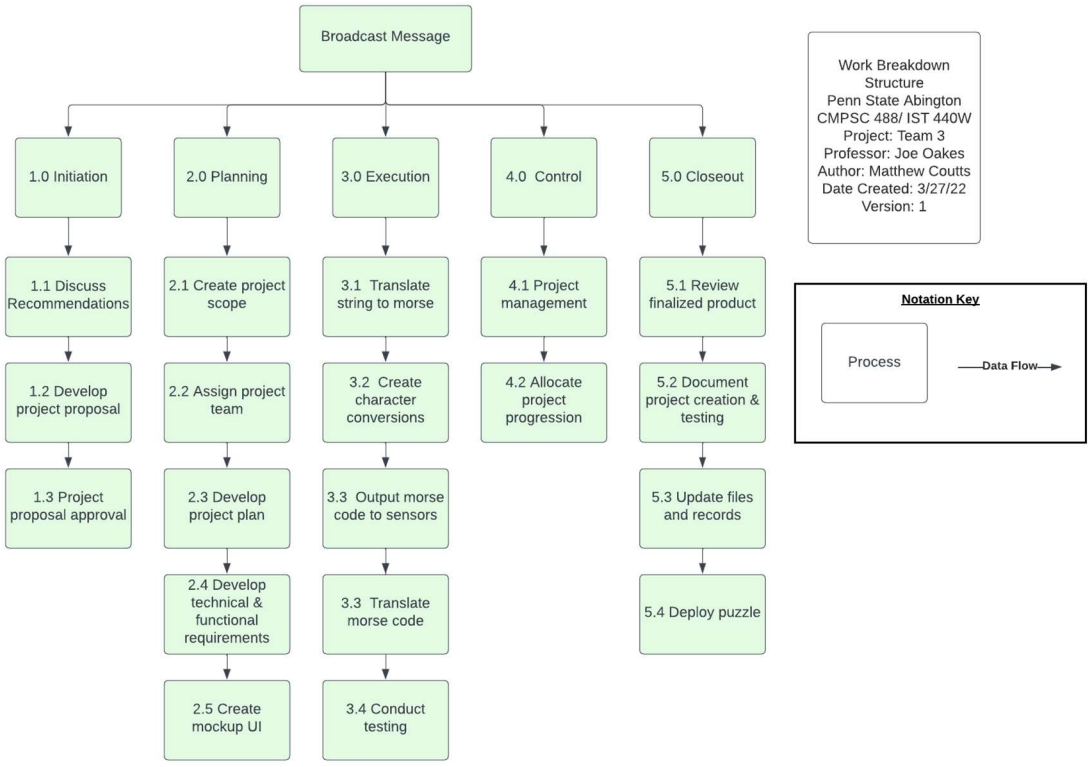


Figure 26 Broadcast Message Work Breakdown Structure

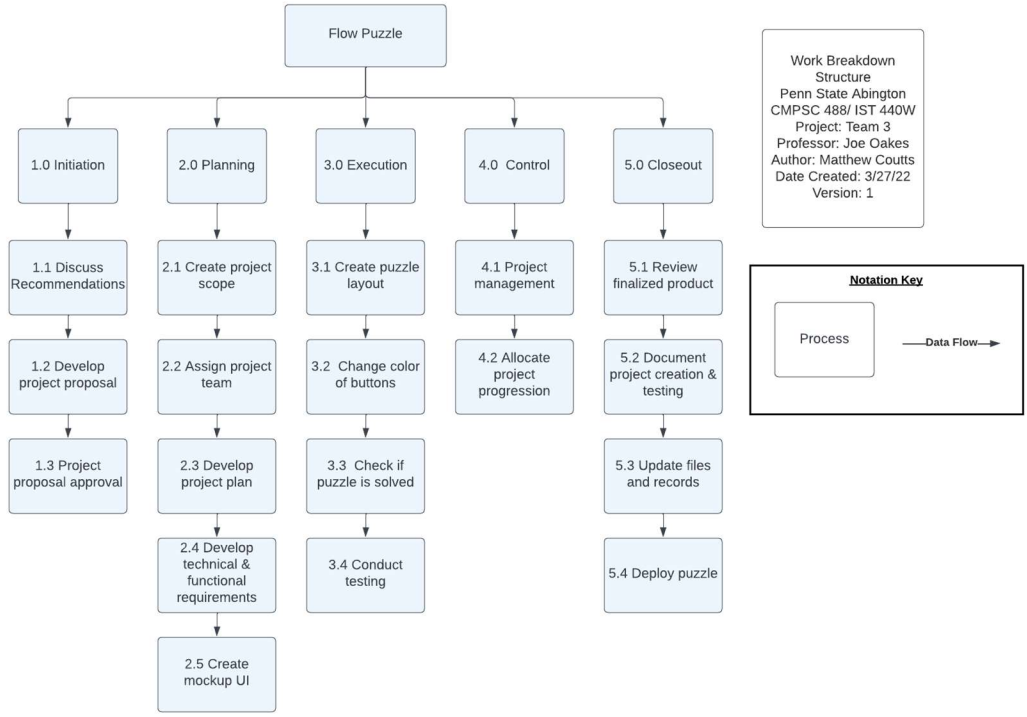


Figure 27 Flow Puzzle Work Breakdown Structure

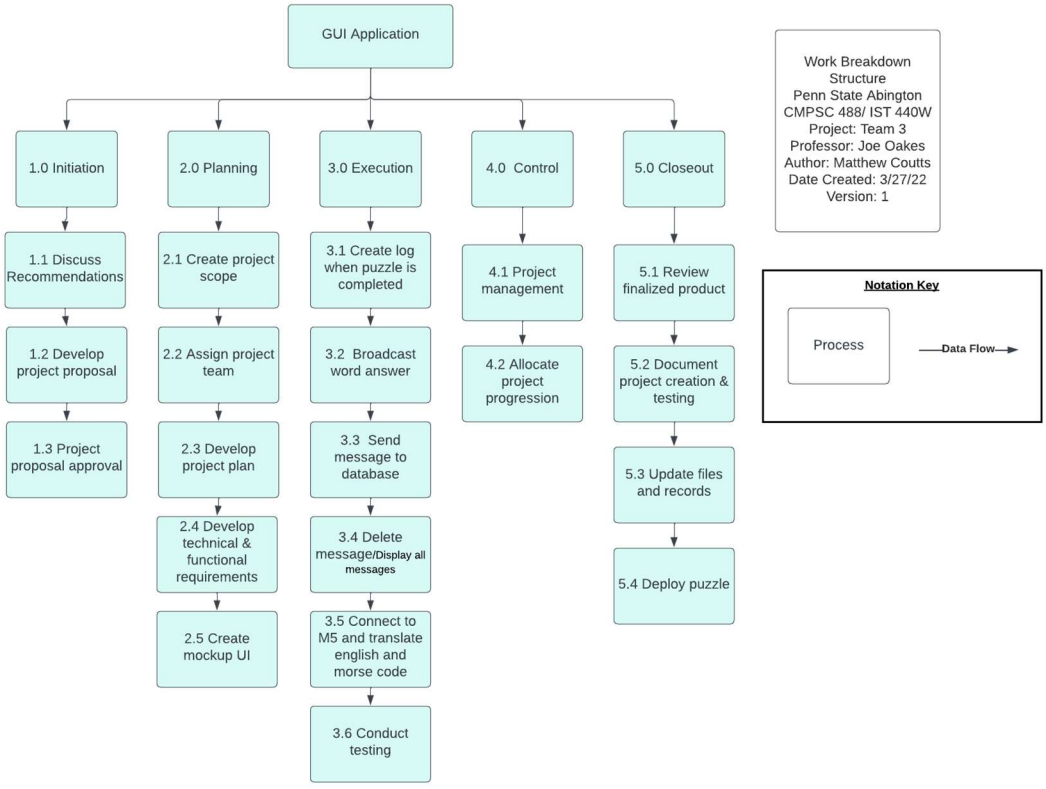


Figure 28 GUI Application Work Breakdown Structure

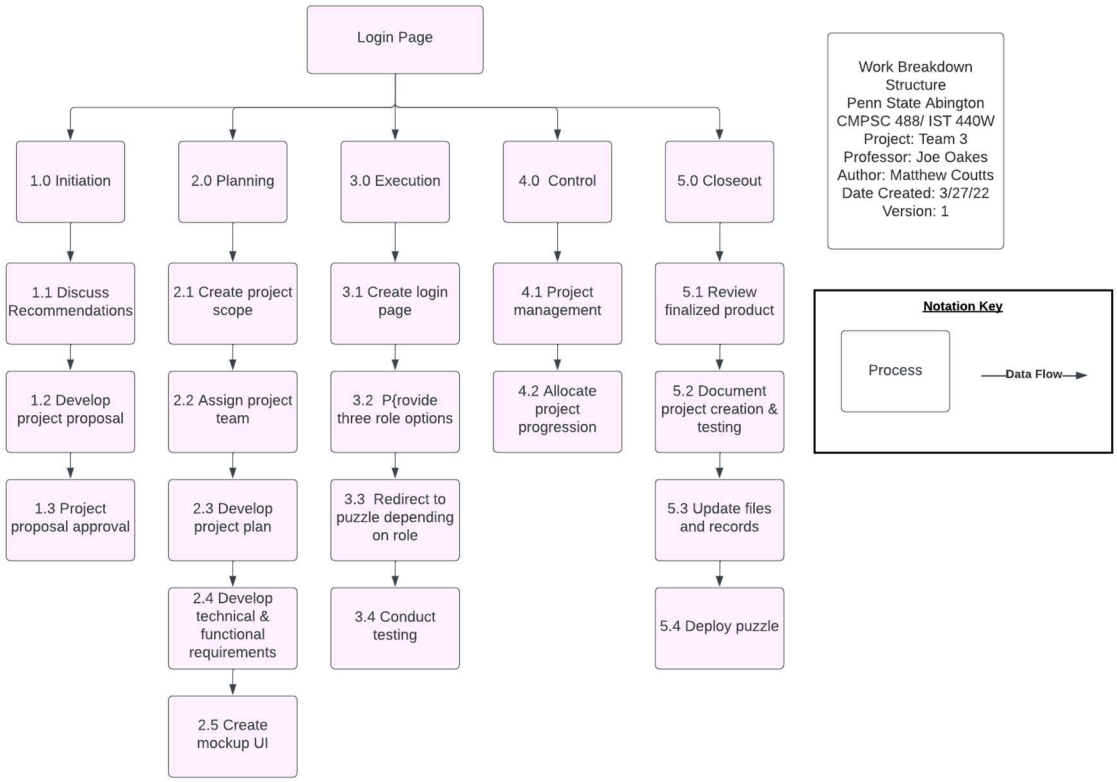


Figure 29 Login Page Work Breakdown Structure

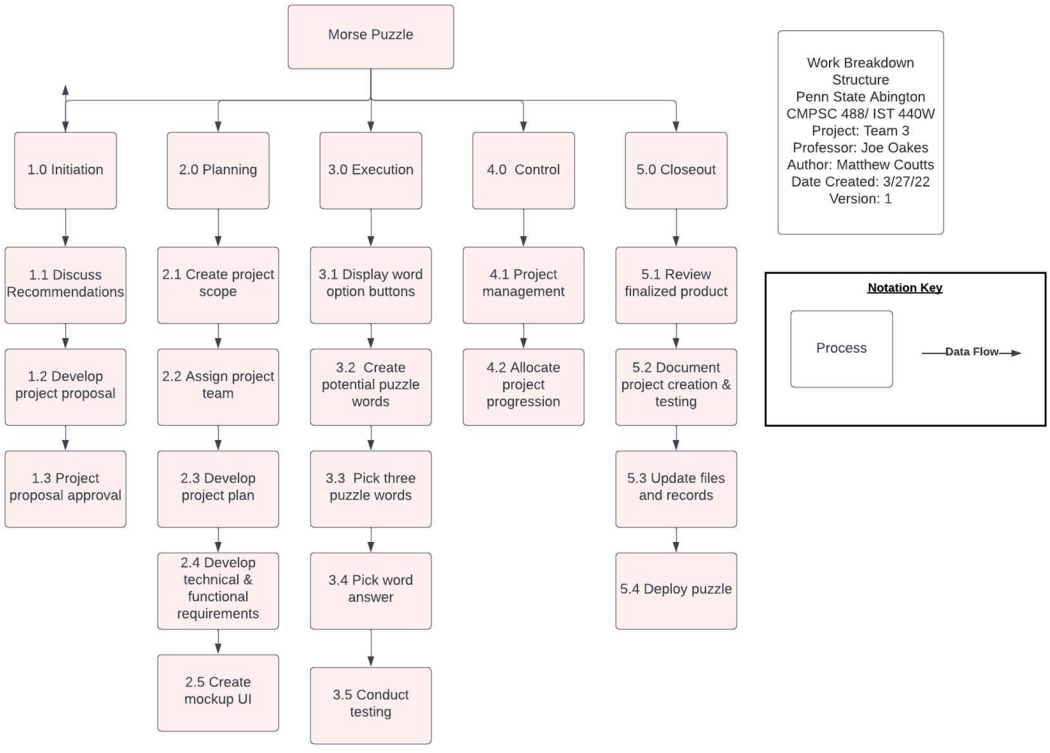


Figure 30 Morse Puzzle Work Breakdown Structure

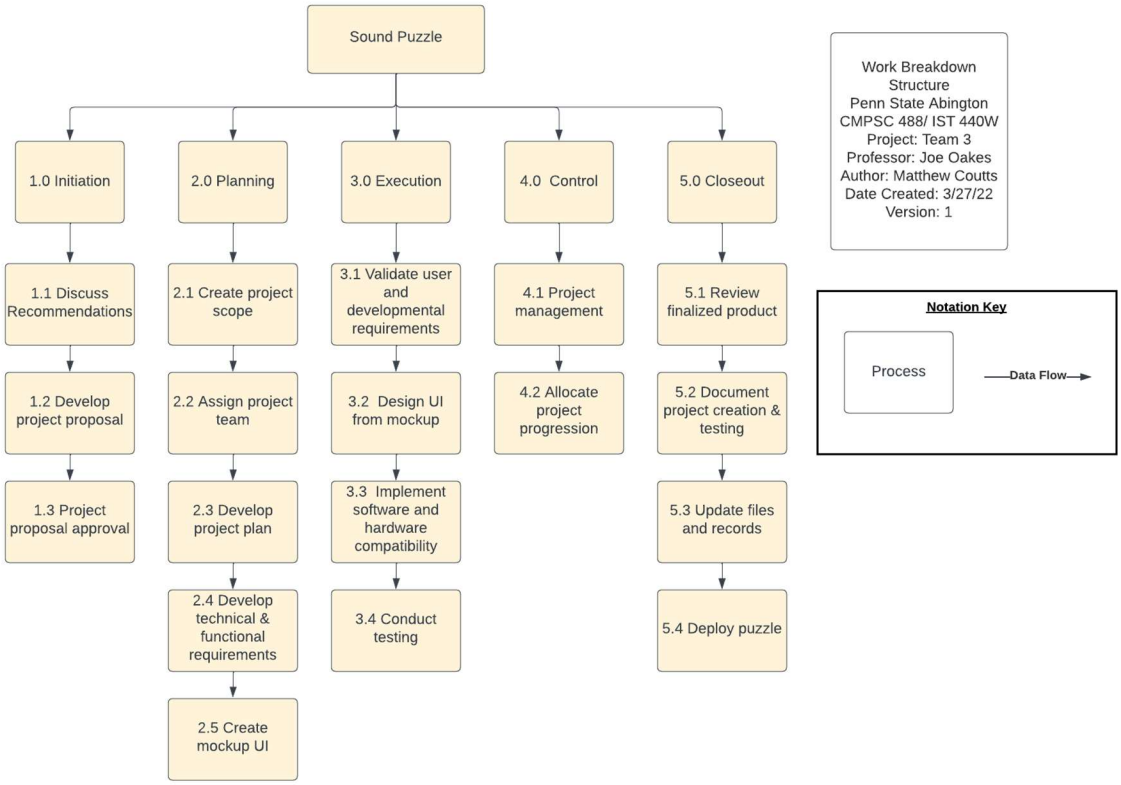


Figure 31 Sound Puzzle Work Breakdown Structure

19.0 Kanban Board

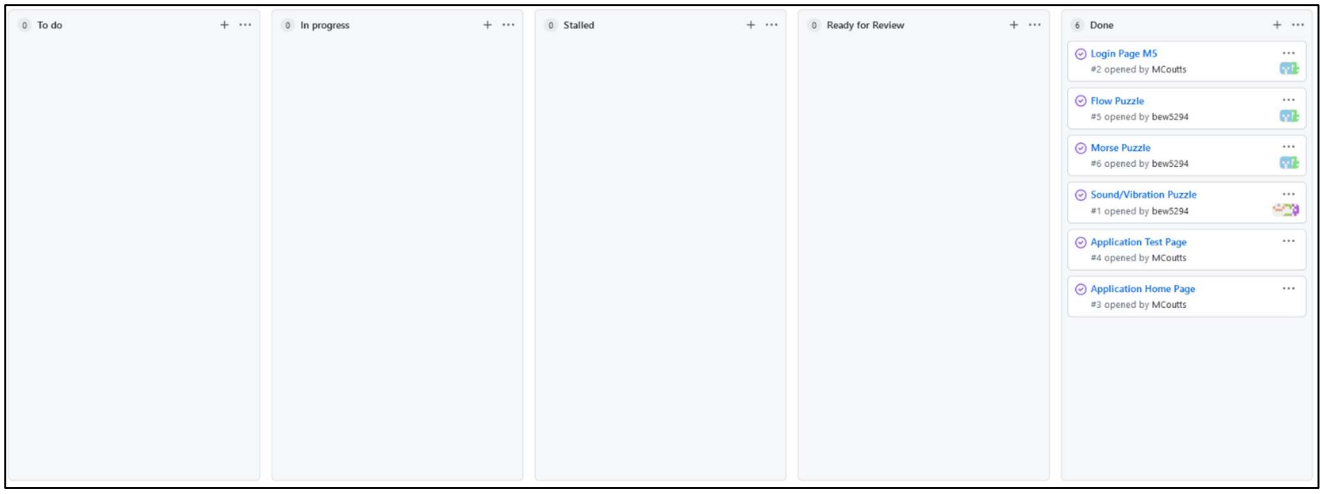


Figure 32: GitHub Project Board

20.0 Code Repository Activity Graphs

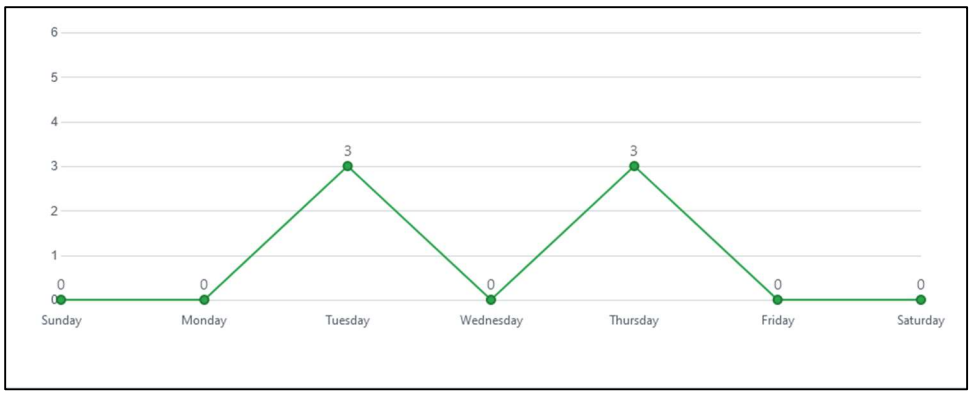


Figure 33: Sprint 1 Code Repository Activity: GitHub (GitHub)

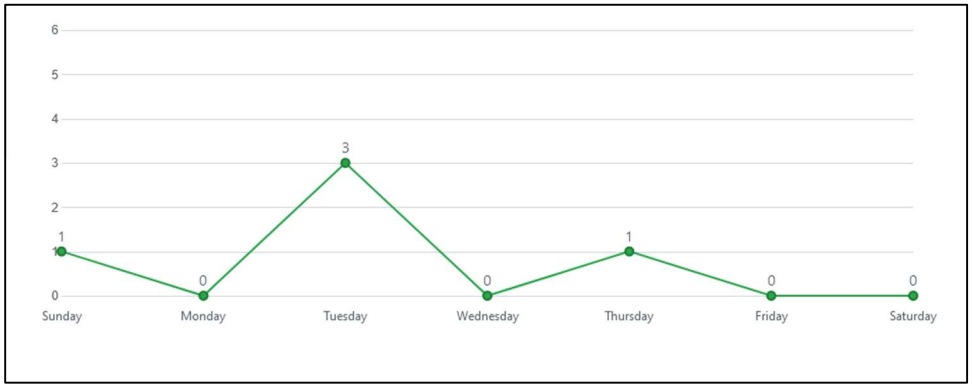


Figure 34: Sprint 2 Code Repository Activity: GitHub (GitHub)

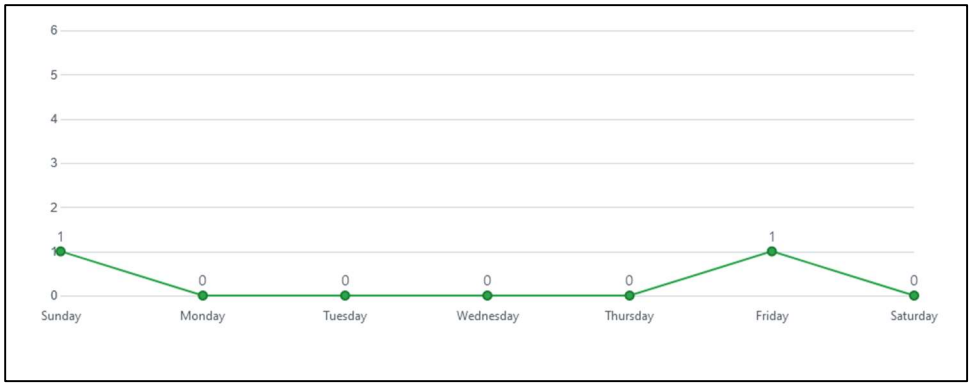


Figure 35: Sprint 3 Code Repository Activity: GitHub (GitHub)

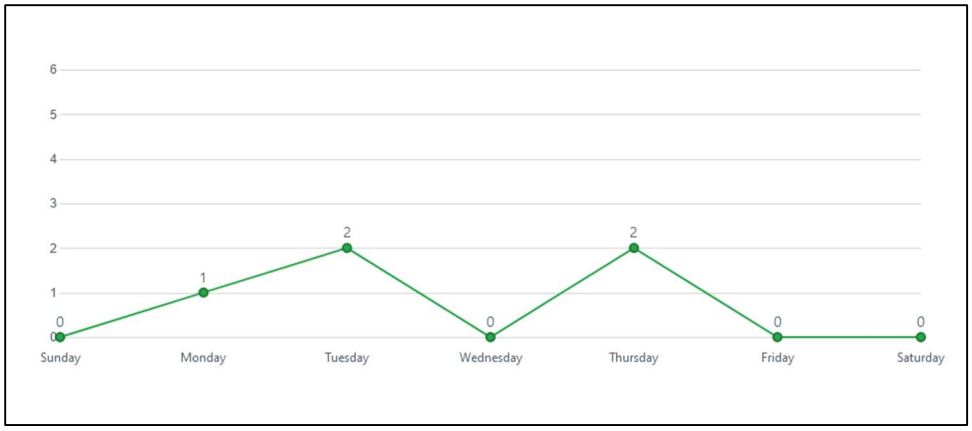


Figure 36: Sprint 4 Code Repository Activity: GitHub (GitHub)

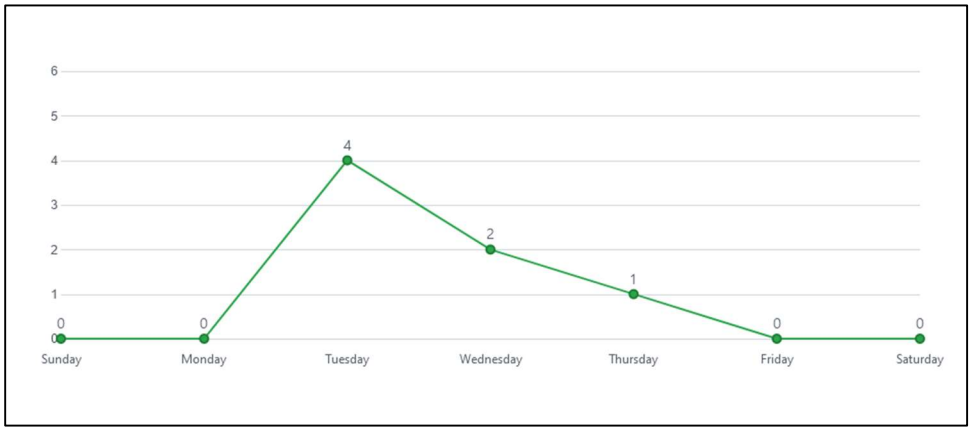


Figure 37: Sprint 5 Code Repository Activity: GitHub (GitHub)

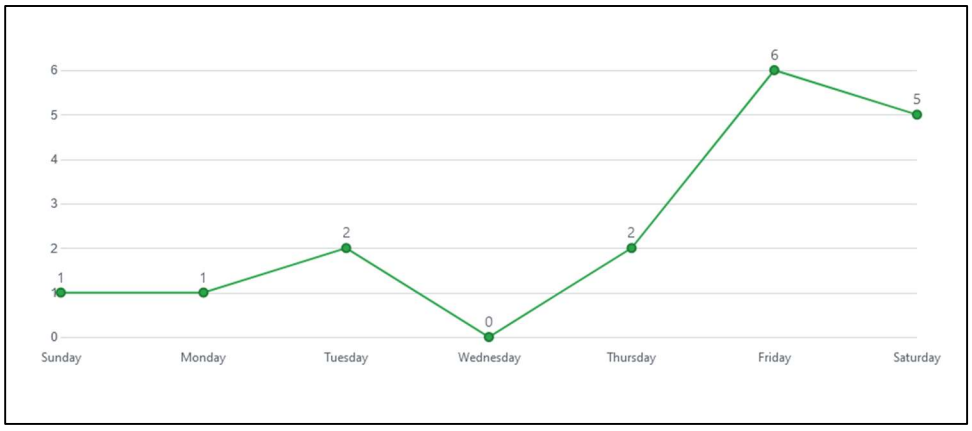


Figure 38: Sprint 6 Code Repository Activity: GitHub (GitHub)

21.0 Burndown Chart

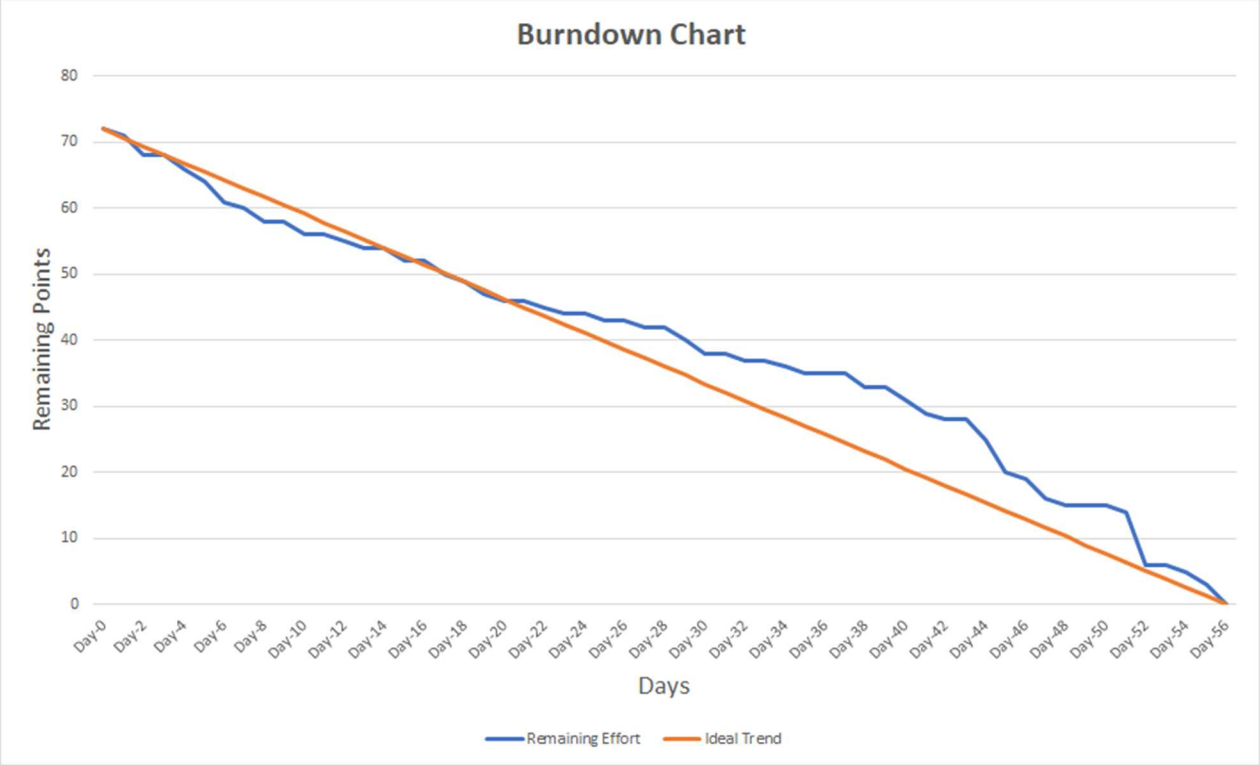


Figure 39: Burndown Chart

21.1 Burndown Analysis

A burndown chart is a line chart drawn between “work remaining” as a vertical axis (Y) and “time” as a horizontal axis (X). This chart is used to track progress and task burn in a sprint. Ideally, the remaining effort curve should be as close as possible to the ideal trend curve. The reason our blue curve goes up after day 20 is because we were having issues with the m5stack. Also, we did not work that much during the spring break. However, at the end we completed all tasks.

22.0 Velocity Charts

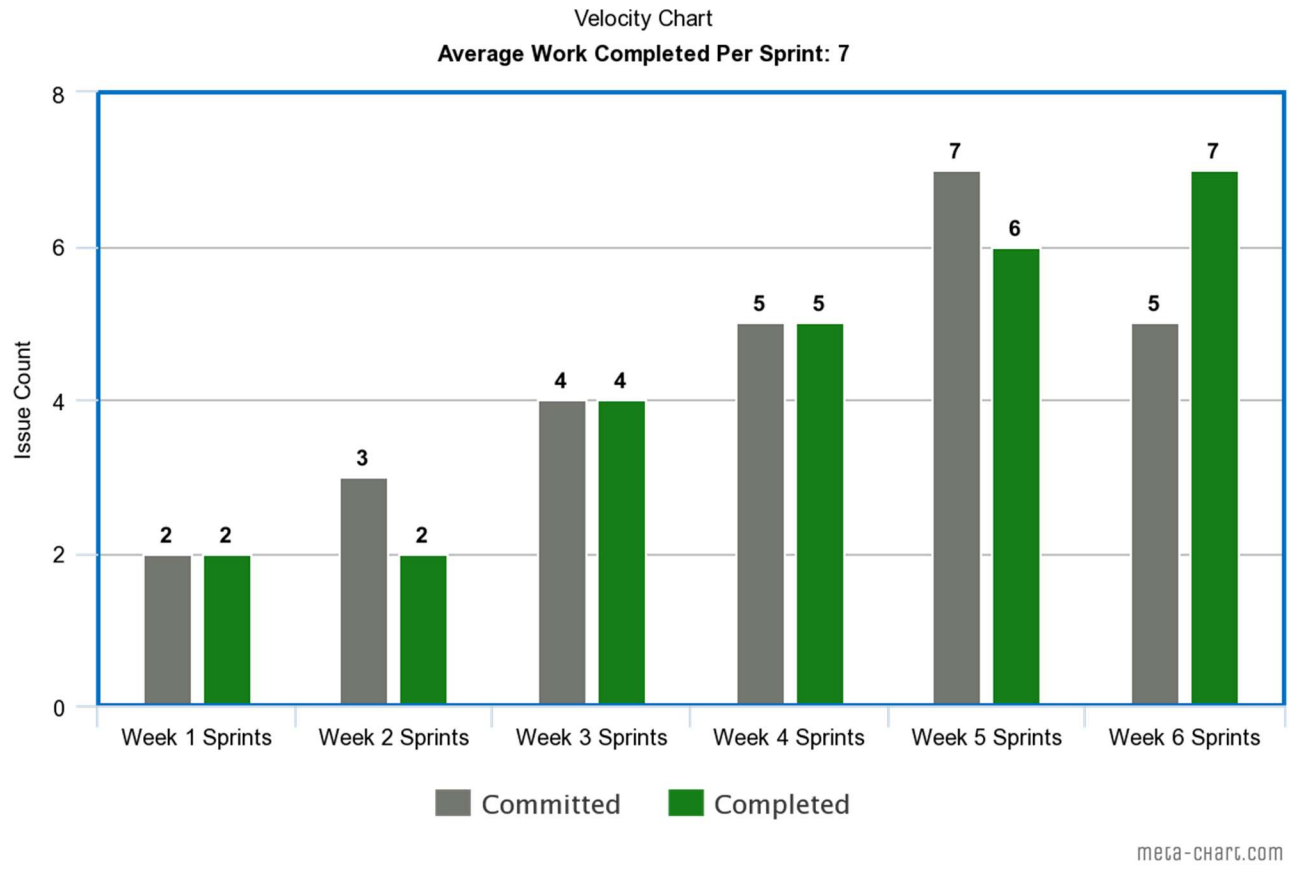


Figure 40: Velocity Chart

22.1 Velocity Analysis

A velocity chart is used as an overview of how much work has been delivered each sprint. This chart is useful for analyzing how quickly our team can work through certain issues. The gray committed bar is the amount of work in the sprint when it started, and the green completed bar indicates how much work was done. Our velocity increased over time as we became more familiar with the project. We had a steady workflow for the most part but in Weeks 2 and 5 we ran into some roadblocks that we resolved in the following weeks.

23.0 Code

23.1 Main.py

```
"""
Project: Team 3 Capstone
Purpose Details: Combination of Login and Puzzle pages
Course: CMPSC 488/ IST 440W
```

```
Author: Bradley Whitebread
Date Developed: 4/1/2022
Last Date Changed: 4/3/2022
Revision: 3
"""

from m5stack import *
from m5stack_ui import *
from uiflow import *
from m5stack import touch
from m5mqtt import M5mqtt
import wifiCfg
import json
import os
import time
import random

screen = M5Screen()
screen.clean_screen()
screen.set_screen_bg_color(0x6998AB)

# wifiCfg.doConnect('Pixel_5294', 'bradley239')
wifiCfg.doConnect('Roveroakes', 'IST888IST888')
# wifiCfg.doConnect('AA Meeting', 'Nittany404')
m5mqtt = M5mqtt('adafruit', 'io.adafruit.com', 1883, 'bew5294',
'aio_pLPP82smfxwykOdHBbsug5RXEBM5', 300)
m5mqtt.start()

"""
Login Page
"""
# Variables
# Intializes UI elements for Login Page
host = '192.168.1.196'
# host = '192.168.234.104'
port = '49153'

puzzleStartTime = None
puzzleEndTime = None

hasLoggedIn = False
roleSelected = False

pickRoleLabel1 = M5Label('Pick Your', x=75, y=29,
                         color=0xffc900, font=FONT_MONT_36, parent=None)
```

```
pickRoleLabel2 = M5Label('Role to Login', x=40, y=73,
                         color=0xffc900, font=FONT_MONT_36, parent=None)

defaultLabel = M5Label('Default', x=30, y=172, color=0x000,
                       font=FONT_MONT_14, parent=None)
maintLabel = M5Label('Maintenance', x=113, y=172,
                     color=0x000, font=FONT_MONT_14, parent=None)
adminLabel = M5Label('Admin', x=240, y=172, color=0x000,
                     font=FONT_MONT_14, parent=None)

downArrowImage1 = M5Img("res/DownArrow.png", x=45, y=195, parent=None)
downArrowImage2 = M5Img("res/DownArrow.png", x=150, y=195, parent=None)
downArrowImage3 = M5Img("res/DownArrow.png", x=255, y=195, parent=None)

# Hides all Login Page UI elements
pickRoleLabel1.set_hidden(True)
pickRoleLabel2.set_hidden(True)

defaultLabel.set_hidden(True)
maintLabel.set_hidden(True)
adminLabel.set_hidden(True)

downArrowImage1.set_hidden(True)
downArrowImage2.set_hidden(True)
downArrowImage3.set_hidden(True)


# Functions
def setUpLoginPage():
    """Displays of UI elements in the Login Page"""
    pickRoleLabel1.set_hidden(False)
    pickRoleLabel2.set_hidden(False)

    wait(1)

    defaultLabel.set_hidden(False)
    maintLabel.set_hidden(False)
    adminLabel.set_hidden(False)

    wait(1)

    downArrowImage1.set_hidden(False)
    downArrowImage2.set_hidden(False)
    downArrowImage3.set_hidden(False)
```

```
def hideLoginPage():
    """Hides all UI elements from the login page"""
    pickRoleLabel1.set_hidden(True)
    pickRoleLabel2.set_hidden(True)

    defaultLabel.set_hidden(True)
    maintLabel.set_hidden(True)
    adminLabel.set_hidden(True)

    downArrowImage1.set_hidden(True)
    downArrowImage2.set_hidden(True)
    downArrowImage3.set_hidden(True)
    pass


def buttonA_wasPressed():
    """The 'default' role is selected. If a role has not been previously selected
the Sound Puzzle will be setup"""
    global roleSelected
    if not roleSelected:
        roleSelected = True
        setUpSoundPuzzle()
    pass
btnA.wasPressed(buttonA_wasPressed)


# Maintenance role selected
def buttonB_wasPressed():
    """The 'maintenance' role is selected. If a role has not been previously
selected the Flow Puzzle will be setup"""
    global roleSelected
    if not roleSelected:
        roleSelected = True
        setUpFlowPuzzle()
    pass
btnB.wasPressed(buttonB_wasPressed)


# Admin role selected
def buttonC_wasPressed():
    """The 'admin' role is selected. If a role has not been previously selected
the Morse Puzzle will be setup"""
    global roleSelected
    if not roleSelected:
        roleSelected = True
```

```
        setUpMorsePuzzle()
    pass
btnC.wasPressed(buttonC_wasPressed)


"""
Sound Puzzle (Astronaut/Default)
"""
# Variables
# All the possible tones that can be played in the Sound puzzle
tones = [
    220, 247, 131, 147, 165, 175, 196, 262, 294, 330,
    349, 392, 233, 139, 156, 185, 208, 277, 311, 370
]

toneLabel = M5Label('0', x=155, y=113, color=0x000,
                    font=FONT_MONT_14, parent=None)
toneLabel.set_hidden(True)

soundOptionBtns = [
    M5Btn(text='Tone 1', x=15,  y=15, w=290, h=60, bg_c=0xFFFFFF,
          text_c=0x000000, font=FONT_MONT_44, parent=None),
    M5Btn(text='Tone 2', x=15,  y=90, w=290, h=60, bg_c=0xFFFFFF,
          text_c=0x000000, font=FONT_MONT_44, parent=None),
    M5Btn(text='Tone 3', x=15, y=165, w=290, h=60, bg_c=0xFFFFFF,
          text_c=0x000000, font=FONT_MONT_44, parent=None)
]
soundOptionBtns[0].set_hidden(True)
soundOptionBtns[1].set_hidden(True)
soundOptionBtns[2].set_hidden(True)

toneOptions = []

toneAnswer = None

# Functions
def setUpSoundPuzzle():
    """Sets up and runs the sound puzzle. Hides the login page then plays three
unique tones then 1 repeated tone.
       The user must decide which of the three tones was repeated"""
    global toneAnswer, toneOptions, toneLabel, puzzleStartTime
    hideLoginPage()                                    # hides the login
page
    wait(1)
```

```
    toneOptions = pickToneOptions()                         # picks three tones
from the tone array
    toneAnswer = toneOptions[int(random.randint(1, 3) - 1)] # picks the correct
tone from the selected tone options
    puzzleStartTime = time.ticks_ms()                       # starts tracking how
long it takes to complete the puzzle
    playTones(toneOptions)                                  # plays the three
tone options
    wait(1)
    playToneAnswer(toneAnswer)                              # plays the selected
tone answer
    displayToneOptions()                                    # displays the tone
options buttons
    pass


def pickToneOptions():
    """Picks three unique tones from the tones array"""
    global tones
    randIndex1 = random.randint(1, len(tones))
    randIndex2 = random.randint(1, len(tones))
    while randIndex1 == randIndex2:
        randIndex2 = random.randint(1, len(tones))

    randIndex3 = random.randint(1, len(tones))
    while randIndex3 == randIndex1 or randIndex3 == randIndex2:
        randIndex3 = random.randint(1, len(tones))

    toneOptions = [                 # fills toneOptions array with the three
unique indices
        tones[int(randIndex1 - 1)], # from the from the tones array
        tones[int(randIndex2 - 1)],
        tones[int(randIndex3 - 1)]
    ]
    return toneOptions


def playTones(toneOptions):
    """Plays the tone options passed to the function"""
    global toneLabel
    toneLabel.set_text('1')
    toneLabel.set_hidden(False)
    speaker.playTone(toneOptions[0], 1) # plays the first tone option
    toneLabel.set_hidden(True)
```

```
    speaker.playTone(1, 1/16)

    toneLabel.set_text('2')
    toneLabel.set_hidden(False)
    speaker.playTone(toneOptions[1], 1) # plays the second tone option
    toneLabel.set_hidden(True)

    speaker.playTone(1, 1/16)

    toneLabel.set_text('3')
    toneLabel.set_hidden(False)
    speaker.playTone(toneOptions[2], 1) # plays the third tone option
    toneLabel.set_hidden(True)
    pass


def playToneAnswer(toneAnswer):
    """Plays the tone answer passed to the function"""
    global toneLabel
    toneLabel.set_text('?')
    toneLabel.set_hidden(False)
    speaker.playTone(toneAnswer, 1)
    toneLabel.set_hidden(True)
    pass


def displayToneOptions():
    """Displays the three tone option buttons"""
    soundOptionBtns[0].set_hidden(False)
    soundOptionBtns[1].set_hidden(False)
    soundOptionBtns[2].set_hidden(False)
    pass


def tone1BtnPressed():
    """Checks if the tone 1 is the correct answer. If it is it posts a log to the
database saying the Sound Puzzle was solved.
       If not the Sound Puzzle is reset for another try with new tones"""
    global toneAnswer, toneOptions, puzzleStartTime, puzzleEndTime, host, port,
hasLoggedIn
    if toneAnswer == toneOptions[0]:              # Checks if tone 1 is the
correct answer to the puzzle
        screen.set_screen_bg_color(0x99ff99)
        puzzleEndTime = time.ticks_ms()           # finished tracking how long
it takes to complete the puzzle
```

```
        duration = puzzleEndTime - puzzleStartTime # calculates puzzle duration
        m5mqtt.publish('bew5294/feeds/capstone-morse-communicator', 'default')
        hasLoggedIn = True
        try:
            req = urequests.request(
                method='POST',
                url='http://' + host + ':' + port + '/logs',
                json={
                    'app': 'Sound Puzzle',
                    'status': 'success',
                    'duration':str(duration)
                }
            )
        except:
            pass
    else:
        screen.set_screen_bg_color(0xffcccc)           # sets the screen red if
tone 3 is the incorrect answer
        wait(1)
        setUpSoundPuzzle()                             # also resets the puzzle
to retry with new tones
    pass
soundOptionBtns[0].pressed(tone1BtnPressed)


def tone2BtnPressed():
    """Checks if the tone 2 is the correct answer. If it is it posts a log to the
database saying the Sound Puzzle was solved.
       If not the Sound Puzzle is reset for another try with new tones"""
    global toneAnswer, toneOptions, puzzleStartTime, puzzleEndTime, host, port,
hasLoggedIn
    if toneAnswer == toneOptions[1]:               # Checks if tone 2 is the
correct answer to the puzzle
        screen.set_screen_bg_color(0x99ff99)
        puzzleEndTime = time.ticks_ms()            # finished tracking how long
it takes to complete the puzzle
        duration = puzzleEndTime - puzzleStartTime # calculates puzzle duration
        m5mqtt.publish('bew5294/feeds/capstone-morse-communicator', 'default')
        hasLoggedIn = True
        try:
            req = urequests.request(
                method='POST',
                url='http://' + host + ':' + port + '/logs',
                json={
                    'app': 'Sound Puzzle',
```

```
                    'status': 'success',
                    'duration':str(duration)
                }
            )
        except:
            pass
    else:
        screen.set_screen_bg_color(0xffcccc)          # sets the screen red if
tone 3 is the incorrect answer
        wait(1)
        setUpSoundPuzzle()                            # also resets the puzzle
to retry with new tones
    pass
soundOptionBtns[1].pressed(tone2BtnPressed)


def tone3BtnPressed():
    """Checks if the tone 3 is the correct answer. If it is it posts a log to the
database saying the Sound Puzzle was solved.
       If not the Sound Puzzle is reset for another try with new tones"""
    global toneAnswer, toneOptions, puzzleStartTime, puzzleEndTime, host, port,
hasLoggedIn
    if toneAnswer == toneOptions[2]:                  # Checks if tone 3 is the
correct answer to the puzzle
        screen.set_screen_bg_color(0x99ff99)          # sets the screen red if
tone 3 is the incorrect answer
        puzzleEndTime = time.ticks_ms()               # finished tracking how
long it takes to complete the puzzle
        duration = puzzleEndTime - puzzleStartTime    # calculates puzzle
duration
        m5mqtt.publish('bew5294/feeds/capstone-morse-communicator', 'default')
        hasLoggedIn = True
        try:
            req = urequests.request(                  # posts json to log
database with the puzzle name, status, and duration
                method='POST',
                url='http://' + host + ':' + port + '/logs',
                json={
                    'app': 'Sound Puzzle',
                    'status': 'success',
                    'duration':str(duration)
                }
            )
        except:
            pass
```

```
    else:
        screen.set_screen_bg_color(0xffcccc)          # sets the screen red if
tone 3 is the incorrect answer
        wait(1)
        setUpSoundPuzzle()                            # also resets the puzzle
to retry with new tones
    pass
soundOptionBtns[2].pressed(tone3BtnPressed)



"""
Flow Puzzle (Maintenance)
"""
# Variables
lastPressed = 'black' # lastPressed is used to store the string of the last
pressed root node
notReleased = True
color = [0, 0, 0]

# added None padding to all 2D arrays for easier index accessing
# declares a 2D array that is the solution to the flow puzzle
solution = [
    [None, None, None, None, None, None],
    [None, 'indigo', 'indigo', 'indigo', 'red',    'red',    'green'],
    [None, 'indigo', 'orange', 'indigo', 'yellow', 'red',    'green'],
    [None, 'indigo', 'orange', 'indigo', 'yellow', 'red',    'green'],
    [None, 'indigo', 'orange', 'indigo', 'yellow', 'blue',   'orange'],
    [None, 'indigo', 'orange', 'blue',   'blue',   'blue',   'orange'],
    [None, 'indigo', 'orange', 'orange', 'orange', 'orange', 'orange']
]

# declares a 2D array that tracks the current position of the flow puzzle
btnTagMatrix = [
    [None, None, None, None, None, None],
    [None, 'black',  'black',  'black',  'red',    'black', 'green'],
    [None, 'black',  'orange', 'black',  'yellow', 'black', 'black'],
    [None, 'black',  'black',  'black',  'black',  'red',   'green'],
    [None, 'black',  'black',  'indigo', 'yellow', 'blue',  'orange'],
    [None, 'black',  'black',  'blue',   'black',  'black', 'black'],
    [None, 'indigo', 'black',  'black',  'black',  'black', 'black']
]

# declares a 2D array of the buttons in the flow puzzle
btnMatrix = [
```

```
    [None, None, None, None, None, None],
    [
        None,
        M5Btn(text='o', x=40, y=0, w=40, h=40, bg_c=0x000000,
              text_c=0x000000, font=FONT_MONT_14, parent=None),
        M5Btn(text='o', x=80, y=0, w=40, h=40, bg_c=0x000000,
              text_c=0x000000, font=FONT_MONT_14, parent=None),
        M5Btn(text='o', x=120, y=0, w=40, h=40, bg_c=0x000000,
              text_c=0x000000, font=FONT_MONT_14, parent=None),
        M5Btn(text='o', x=160, y=0, w=40, h=40, bg_c=0xff0000,
              text_c=0x000000, font=FONT_MONT_14, parent=None),
        M5Btn(text='o', x=200, y=0, w=40, h=40, bg_c=0x000000,
              text_c=0x000000, font=FONT_MONT_14, parent=None),
        M5Btn(text='o', x=240, y=0, w=40, h=40, bg_c=0x006600,
              text_c=0x000000, font=FONT_MONT_14, parent=None)
    ],
    [
        None,
        M5Btn(text='o', x=40, y=40, w=40, h=40, bg_c=0x000000,
              text_c=0x000000, font=FONT_MONT_14, parent=None),
        M5Btn(text='o', x=80, y=40, w=40, h=40, bg_c=0xff9900,
              text_c=0x000000, font=FONT_MONT_14, parent=None),
        M5Btn(text='o', x=120, y=40, w=40, h=40, bg_c=0x000000,
              text_c=0x000000, font=FONT_MONT_14, parent=None),
        M5Btn(text='o', x=160, y=40, w=40, h=40, bg_c=0xffff00,
              text_c=0x000000, font=FONT_MONT_14, parent=None),
        M5Btn(text='o', x=200, y=40, w=40, h=40, bg_c=0x000000,
              text_c=0x000000, font=FONT_MONT_14, parent=None),
        M5Btn(text='o', x=240, y=40, w=40, h=40, bg_c=0x000000,
              text_c=0x000000, font=FONT_MONT_14, parent=None)
    ],
    [
        None,
        M5Btn(text='o', x=40, y=80, w=40, h=40, bg_c=0x000000,
              text_c=0x000000, font=FONT_MONT_14, parent=None),
        M5Btn(text='o', x=80, y=80, w=40, h=40, bg_c=0x000000,
              text_c=0x000000, font=FONT_MONT_14, parent=None),
        M5Btn(text='o', x=120, y=80, w=40, h=40, bg_c=0x000000,
              text_c=0x000000, font=FONT_MONT_14, parent=None),
        M5Btn(text='o', x=160, y=80, w=40, h=40, bg_c=0x000000,
              text_c=0x000000, font=FONT_MONT_14, parent=None),
        M5Btn(text='o', x=200, y=80, w=40, h=40, bg_c=0xff0000,
              text_c=0x000000, font=FONT_MONT_14, parent=None),
        M5Btn(text='o', x=240, y=80, w=40, h=40, bg_c=0x006600,
              text_c=0x000000, font=FONT_MONT_14, parent=None)
```

```
    ],
    [
        None,
        M5Btn(text='o', x=40, y=120, w=40, h=40, bg_c=0x000000,
              text_c=0x000000, font=FONT_MONT_14, parent=None),
        M5Btn(text='o', x=80, y=120, w=40, h=40, bg_c=0x000000,
              text_c=0x000000, font=FONT_MONT_14, parent=None),
        M5Btn(text='o', x=120, y=120, w=40, h=40, bg_c=0x3333ff,
              text_c=0x000000, font=FONT_MONT_14, parent=None),
        M5Btn(text='o', x=160, y=120, w=40, h=40, bg_c=0xffff00,
              text_c=0x000000, font=FONT_MONT_14, parent=None),
        M5Btn(text='o', x=200, y=120, w=40, h=40, bg_c=0x33ffff,
              text_c=0x000000, font=FONT_MONT_14, parent=None),
        M5Btn(text='o', x=240, y=120, w=40, h=40, bg_c=0xff9900,
              text_c=0x000000, font=FONT_MONT_14, parent=None)
    ],
    [
        None,
        M5Btn(text='o', x=40, y=160, w=40, h=40, bg_c=0x000000,
              text_c=0x000000, font=FONT_MONT_14, parent=None),
        M5Btn(text='o', x=80, y=160, w=40, h=40, bg_c=0x000000,
              text_c=0x000000, font=FONT_MONT_14, parent=None),
        M5Btn(text='o', x=120, y=160, w=40, h=40, bg_c=0x33ffff,
              text_c=0x000000, font=FONT_MONT_14, parent=None),
        M5Btn(text='o', x=160, y=160, w=40, h=40, bg_c=0x000000,
              text_c=0x000000, font=FONT_MONT_14, parent=None),
        M5Btn(text='o', x=200, y=160, w=40, h=40, bg_c=0x000000,
              text_c=0x000000, font=FONT_MONT_14, parent=None),
        M5Btn(text='o', x=240, y=160, w=40, h=40, bg_c=0x000000,
              text_c=0x000000, font=FONT_MONT_14, parent=None)
    ],
    [
        None,
        M5Btn(text='o', x=40, y=200, w=40, h=40, bg_c=0x3333ff,
              text_c=0x000000, font=FONT_MONT_14, parent=None),
        M5Btn(text='o', x=80, y=200, w=40, h=40, bg_c=0x000000,
              text_c=0x000000, font=FONT_MONT_14, parent=None),
        M5Btn(text='o', x=120, y=200, w=40, h=40, bg_c=0x000000,
              text_c=0x000000, font=FONT_MONT_14, parent=None),
        M5Btn(text='o', x=160, y=200, w=40, h=40, bg_c=0x000000,
              text_c=0x000000, font=FONT_MONT_14, parent=None),
        M5Btn(text='o', x=200, y=200, w=40, h=40, bg_c=0x000000,
              text_c=0x000000, font=FONT_MONT_14, parent=None),
        M5Btn(text='o', x=240, y=200, w=40, h=40, bg_c=0x000000,
              text_c=0x000000, font=FONT_MONT_14, parent=None)
```

```
    ]
]

for row in range(1, len(btnMatrix)):              # hides all the buttons in the
btn matrix
    for col in range(1, len(btnMatrix[row])):
        btnMatrix[row][col].set_hidden(True)


# Functions
def setUpFlowPuzzle():
    """Hides the login page and displays all the buttons in the flow puzzle"""
    global btnMatrix, puzzleStartTime
    hideLoginPage()                               # hides login page
    for row in range(1, len(btnMatrix)):          # shows all the buttons in the
btn matrix
        for col in range(1, len(btnMatrix[row])):
            btnMatrix[row][col].set_hidden(False)
    puzzleStartTime = time.ticks_ms()


def checkFlowPuzzle():
    """Checks if every button in the puzzle is correctly set. There is only one
correct configuration for the puzzle.
       If the puzzle is correct a log is posted to the database saying the Flow
Puzzle was solved"""
    global color, notReleased, lastPressed, btnTagMatrix, puzzleStartTime,
puzzleEndTime, hasLoggedIn
    if btnTagMatrix == solution:                      # checks if btn tag matrix
is equal to the solution matrix
        puzzleEndTime = time.ticks_ms()               # finished tracking how
long it takes to complete the puzzle
        duration = puzzleEndTime - puzzleStartTime    # calculates puzzle
duration
        m5mqtt.publish('bew5294/feeds/capstone-morse-communicator',
'maintenance')
        hasLoggedIn = True
        try:
            req = urequests.request(                  # posts json to log
database with the puzzle name, status, and duration
                method='POST',
                url='http://' + host + ':' + port + '/logs',
                json={
                    'app':'Flow Puzzle',
                    'status':'completed',
```

```
                    'duration':str(duration)
                }
            )
        except:
            pass
    pass


def changeBtnColor(row, col):
    """Changes the color of the button at the inputted coordinates"""
    global color, notReleased, lastPressed, btnMatrix, btnTagMatrix
    if notReleased:                                         # checks that the
screen has been continuously pressed since touching a root node
        btnMatrix[row][col].set_bg_color(                   # sets the button
color to the color of the last pressed root node
            (color[0] << 16) | (color[1] << 8) | color[2])
        btnMatrix[row][col].set_btn_text_color(             # sets the button text
color to the color of the last pressed root node
            (color[0] << 16) | (color[1] << 8) | color[2])
        btnTagMatrix[row][col] = lastPressed                # sets the button tag
to the last pressed root node color
    pass


def setLastPressed(colorName):
    """Sets the current color and lastPressed variable to that of the inputted
string"""
    global color, lastPressed, notReleased

    notReleased = True

    if colorName == 'red':
        color = [255, 0, 0]
        lastPressed = 'red'
        pass

    elif colorName == 'orange':
        color = [255, 153, 0]
        lastPressed = 'orange'
        pass

    elif colorName == 'yellow':
        color = [255, 255, 0]
        lastPressed = 'yellow'
        pass
```

```
    elif colorName == 'green':
        color = [0, 102, 0]
        lastPressed = 'green'
        pass

    elif colorName == 'blue':
        color = [51, 255, 255]
        lastPressed = 'blue'
        pass

    elif colorName == 'indigo':
        color = [51, 51, 255]
        lastPressed = 'indigo'
        pass
    pass

# root buttons (root refers to one end of a line that cannot be changed)
def redRootPressed():
    """Sets last pressed to 'red'"""
    setLastPressed('red')
    pass
btnMatrix[1][4].pressed(redRootPressed)
btnMatrix[3][5].pressed(redRootPressed)


def orangeRootPressed():
    """Sets last pressed to 'orange'"""
    setLastPressed('orange')
    pass
btnMatrix[2][2].pressed(orangeRootPressed)
btnMatrix[4][6].pressed(orangeRootPressed)


def yellowRootPressed():
    """Sets last pressed to 'yellow'"""
    setLastPressed('yellow')
    pass
btnMatrix[2][4].pressed(yellowRootPressed)
btnMatrix[4][4].pressed(yellowRootPressed)


def greenRootPressed():
    """Sets last pressed to 'green'"""
    setLastPressed('green')
```

```
    pass
btnMatrix[1][6].pressed(greenRootPressed)
btnMatrix[3][6].pressed(greenRootPressed)


def blueRootPressed():
    """Sets last pressed to 'blue'"""
    setLastPressed('blue')
    pass
btnMatrix[4][5].pressed(blueRootPressed)
btnMatrix[5][3].pressed(blueRootPressed)


def indigoRootPressed():
    """Sets last pressed to 'indigo'"""
    setLastPressed('indigo')
    pass
btnMatrix[4][3].pressed(indigoRootPressed)
btnMatrix[6][1].pressed(indigoRootPressed)


# non-root buttons (the color of these buttons can be changed)
def btn11Pressed():
    changeBtnColor(1, 1)
    pass
btnMatrix[1][1].pressed(btn11Pressed)


def btn12Pressed():
    changeBtnColor(1, 2)
    pass
btnMatrix[1][2].pressed(btn12Pressed)


def btn13Pressed():
    changeBtnColor(1, 3)
    pass
btnMatrix[1][3].pressed(btn13Pressed)


def btn15Pressed():
    changeBtnColor(1, 5)
    pass
btnMatrix[1][5].pressed(btn15Pressed)
```

```
def btn21Pressed():
    changeBtnColor(2, 1)
    pass
btnMatrix[2][1].pressed(btn21Pressed)


def btn23Pressed():
    changeBtnColor(2, 3)
    pass
btnMatrix[2][3].pressed(btn23Pressed)


def btn25Pressed():
    changeBtnColor(2, 5)
    pass
btnMatrix[2][5].pressed(btn25Pressed)


def btn26Pressed():
    changeBtnColor(2, 6)
    pass
btnMatrix[2][6].pressed(btn26Pressed)


def btn31Pressed():
    changeBtnColor(3, 1)
    pass
btnMatrix[3][1].pressed(btn31Pressed)


def btn32Pressed():
    changeBtnColor(3, 2)
    pass
btnMatrix[3][2].pressed(btn32Pressed)


def btn33Pressed():
    changeBtnColor(3, 3)
    pass
btnMatrix[3][3].pressed(btn33Pressed)


def btn34Pressed():
    changeBtnColor(3, 4)
    pass
btnMatrix[3][4].pressed(btn34Pressed)
```

```
def btn41Pressed():
    changeBtnColor(4, 1)
    pass
btnMatrix[4][1].pressed(btn41Pressed)


def btn42Pressed():
    changeBtnColor(4, 2)
    pass
btnMatrix[4][2].pressed(btn42Pressed)


def btn51Pressed():
    changeBtnColor(5, 1)
    pass
btnMatrix[5][1].pressed(btn51Pressed)


def btn52Pressed():
    changeBtnColor(5, 2)
    pass
btnMatrix[5][2].pressed(btn52Pressed)


def btn54Pressed():
    changeBtnColor(5, 4)
    pass
btnMatrix[5][4].pressed(btn54Pressed)


def btn55Pressed():
    changeBtnColor(5, 5)
    pass
btnMatrix[5][5].pressed(btn55Pressed)


def btn56Pressed():
    changeBtnColor(5, 6)
    pass
btnMatrix[5][6].pressed(btn56Pressed)


def btn62Pressed():
    changeBtnColor(6, 2)
```

```
    pass
btnMatrix[6][2].pressed(btn62Pressed)


def btn63Pressed():
    changeBtnColor(6, 3)
    pass
btnMatrix[6][3].pressed(btn63Pressed)


def btn64Pressed():
    changeBtnColor(6, 4)
    pass
btnMatrix[6][4].pressed(btn64Pressed)


def btn65Pressed():
    changeBtnColor(6, 5)
    pass
btnMatrix[6][5].pressed(btn65Pressed)


def btn66Pressed():
    changeBtnColor(6, 6)
    pass
btnMatrix[6][6].pressed(btn66Pressed)


def btnReleased():
    """When a button is released the status of notReleased is updated and the
puzzle is checked if it is solved"""
    global notReleased
    notReleased = touch.status() # updates the status of notReleased (False if
the screen has been released)
    checkFlowPuzzle()            # checks if the puzzle has been solved
    pass
btnMatrix[1][1].released(btnReleased)
btnMatrix[1][2].released(btnReleased)
btnMatrix[1][3].released(btnReleased)
btnMatrix[1][4].released(btnReleased)
btnMatrix[1][5].released(btnReleased)
btnMatrix[1][6].released(btnReleased)

btnMatrix[2][1].released(btnReleased)
btnMatrix[2][2].released(btnReleased)
btnMatrix[2][3].released(btnReleased)
```

```
btnMatrix[2][4].released(btnReleased)
btnMatrix[2][5].released(btnReleased)
btnMatrix[2][6].released(btnReleased)

btnMatrix[3][1].released(btnReleased)
btnMatrix[3][2].released(btnReleased)
btnMatrix[3][3].released(btnReleased)
btnMatrix[3][4].released(btnReleased)
btnMatrix[3][5].released(btnReleased)
btnMatrix[3][6].released(btnReleased)

btnMatrix[4][1].released(btnReleased)
btnMatrix[4][2].released(btnReleased)
btnMatrix[4][3].released(btnReleased)
btnMatrix[4][4].released(btnReleased)
btnMatrix[4][5].released(btnReleased)
btnMatrix[4][6].released(btnReleased)

btnMatrix[5][1].released(btnReleased)
btnMatrix[5][2].released(btnReleased)
btnMatrix[5][3].released(btnReleased)
btnMatrix[5][4].released(btnReleased)
btnMatrix[5][5].released(btnReleased)
btnMatrix[5][6].released(btnReleased)

btnMatrix[6][1].released(btnReleased)
btnMatrix[6][2].released(btnReleased)
btnMatrix[6][3].released(btnReleased)
btnMatrix[6][4].released(btnReleased)
btnMatrix[6][5].released(btnReleased)
btnMatrix[6][6].released(btnReleased)


"""
Morse Puzzle (Admin)
"""
# Variables
morseAnswer = None
morseOptions = []

morseLabel = M5Label('Text', x=0, y=77, color=0x000,
                     font=FONT_MONT_14, parent=None)
morseLabel.set_hidden(True)

morseOptionBtns = [
```

```
    M5Btn(text='Button', x=15,  y=15, w=290, h=60, bg_c=0xFFFFFF,
          text_c=0x000000, font=FONT_MONT_44, parent=None),
    M5Btn(text='Button', x=15,  y=90, w=290, h=60, bg_c=0xFFFFFF,
          text_c=0x000000, font=FONT_MONT_44, parent=None),
    M5Btn(text='Button', x=15, y=165, w=290, h=60, bg_c=0xFFFFFF,
          text_c=0x000000, font=FONT_MONT_44, parent=None)
]
morseOptionBtns[0].set_hidden(True)
morseOptionBtns[1].set_hidden(True)
morseOptionBtns[2].set_hidden(True)


# Functions
def setUpMorsePuzzle():
    """Sets up and runs the Morse Puzzle. Hides the login page then picks and
broadcasts a word in Morse code.
       The user must decode and pick the word's english translation from three
options"""
    global morseOptions, morseAnswer, puzzleStartTime
    hideLoginPage()                                                  # hides the
login page
    morseOptions = pickOptions()                                     # picks
three options from the puzzle words json file
    morseAnswer = morseOptions[int(random.randint(1, 3) - 1)]['word'] # picks the
answer from the three options
    puzzleStartTime = time.ticks_ms()                                # starts
tracking how long it takes to complete the puzzle
    broadcastMessage(morseAnswer)                                    # outputs
the answer in morse code
    displayOptions(morseOptions)                                     # displays
the more option buttons


def pickOptions():
    """Picks three unique words from puzzleWords.json"""
    with open('/sd/puzzleWords.json', 'r') as fs:                    # opens
puzzleWords.json
        allWords = json.loads((fs.read()))                           # loads json
file into allWords dictionary

    wordList = allWords['words']                                     # takes all
entries in the 'words' key from puzzleWords.json
    randIndex1 = random.randint(1, len(wordList))
    randIndex2 = random.randint(1, len(wordList))
    while randIndex1 == randIndex2:
```

```
        randIndex2 = random.randint(1, len(wordList))

    randIndex3 = random.randint(1, len(wordList))
    while randIndex3 == randIndex1 or randIndex3 == randIndex2:
        randIndex3 = random.randint(1, len(wordList))

    morseOptions = [                                            # fills
morseOptions array with the three unique words
        wordList[int(randIndex1 - 1)],
        wordList[int(randIndex2 - 1)],
        wordList[int(randIndex3 - 1)]
    ]
    return morseOptions


def displayOptions(wordOptions):
    """Displays all the morseOption buttons"""
    global morseOptionBtns
    morseOptionBtns[0].set_btn_text((wordOptions[0]['word'])) # sets the button
text to each word
    morseOptionBtns[1].set_btn_text((wordOptions[1]['word']))
    morseOptionBtns[2].set_btn_text((wordOptions[2]['word']))
    morseOptionBtns[0].set_hidden(False)                        # displays all more
options buttons
    morseOptionBtns[1].set_hidden(False)
    morseOptionBtns[2].set_hidden(False)


def morse1BtnPressed():
    """Checks if the first word is the correct translation. If it is a log is
posted to the database saying the Morse Puzzle was solved.
       If not the Morse Puzzle is reset for another try with a new word"""
    global morseOptions, puzzleStartTime, puzzleEndTime, host, port, hasLoggedIn
    if morseAnswer == (morseOptions[0]['word']):
        screen.set_screen_bg_color(0x99ff99)
        puzzleEndTime = time.ticks_ms()                         # finished tracking
how long it takes to complete the puzzle
        duration = puzzleEndTime - puzzleStartTime              # calculates puzzle
duration
        m5mqtt.publish('bew5294/feeds/capstone-morse-communicator', 'admin')
        hasLoggedIn = True
        try:
            req = urequests.request(
                method='POST',
                url='http://' + host + ':' + port + '/logs',
```

```
                json={
                    'app': 'Morse Puzzle',
                    'status': 'success',
                    'duration':str(duration)
                }
            )
        except:
            pass
    else:
        screen.set_screen_bg_color(0xffcccc)                # sets the screen
red if tone 3 is the incorrect answer
        wait(1)
        setUpMorsePuzzle()                                  # also resets the
puzzle to retry with new words
    pass
morseOptionBtns[0].pressed(morse1BtnPressed)


def morse2BtnPressed():
    """Checks if the second word is the correct translation. If it is a log is
posted to the database saying the Morse Puzzle was solved.
       If not the Morse Puzzle is reset for another try with a new word"""
    global morseOptions, puzzleStartTime, puzzleEndTime, host, port, hasLoggedIn
    if morseAnswer == (morseOptions[1]['word']):
        screen.set_screen_bg_color(0x99ff99)
        puzzleEndTime = time.ticks_ms()                     # finished tracking
how long it takes to complete the puzzle
        duration = puzzleEndTime - puzzleStartTime          # calculates puzzle
duration
        m5mqtt.publish('bew5294/feeds/capstone-morse-communicator', 'admin')
        hasLoggedIn = True
        try:
            req = urequests.request(
                method='POST',
                url='http://' + host + ':' + port + '/logs',
                json={
                    'app': 'Sound Puzzle',
                    'status': 'success',
                    'duration':str(duration)
                }
            )
        except:
            pass
    else:
```

```
        screen.set_screen_bg_color(0xffcccc)                # sets the screen
red if tone 3 is the incorrect answer
        wait(1)
        setUpMorsePuzzle()                                  # also resets the
puzzle to retry with new words
    pass
morseOptionBtns[1].pressed(morse2BtnPressed)


def morse3BtnPressed():
    """Checks if the third word is the correct translation. If it is a log is
posted to the database saying the Morse Puzzle was solved.
       If not the Morse Puzzle is reset for another try with a new word"""
    global morseOptions, puzzleStartTime, puzzleEndTime, host, port, hasLoggedIn
    if morseAnswer == (morseOptions[2]['word']):
        screen.set_screen_bg_color(0x99ff99)
        puzzleEndTime = time.ticks_ms()                     # finished tracking
how long it takes to complete the puzzle
        duration = puzzleEndTime - puzzleStartTime          # calculates puzzle
duration
        m5mqtt.publish('bew5294/feeds/capstone-morse-communicator', 'admin')
        hasLoggedIn = True
        try:
            req = urequests.request(
                method='POST',
                url='http://' + host + ':' + port + '/logs',
                json={
                    'app': 'Sound Puzzle',
                    'status': 'success',
                    'duration':str(duration)
                }
            )
        except:
            pass
    else:
        screen.set_screen_bg_color(0xffcccc)                # sets the screen
red if tone 3 is the incorrect answer
        wait(1)
        setUpMorsePuzzle()                                  # also resets the
puzzle to retry with new words
    pass
morseOptionBtns[2].pressed(morse3BtnPressed)


"""
```

```
Broadcasting
"""
# Functions
def initializePuzzleWords():
    """Creates a json file called puzzleWords.json with 20 puzzle words to use in
the Morse Puzzle"""
    with open('/sd/puzzleWords.json', 'w+') as fs:
        fs.write(str(json.dumps({
            'words': [                                                  #
creates a list of dictionaries
                {'word': 'world', 'translation': translateString('world')}, #
creates a dictionary of words with a word key and a key for the morse translation
                {'word': 'house', 'translation': translateString('house')},
                {'word': 'place', 'translation': translateString('place')},
                {'word': 'group', 'translation': translateString('group')},
                {'word': 'flame', 'translation': translateString('flame')},
                {'word': 'clock', 'translation': translateString('clock')},
                {'word': 'point', 'translation': translateString('point')},
                {'word': 'night', 'translation': translateString('night')},
                {'word': 'water', 'translation': translateString('water')},
                {'word': 'thing', 'translation': translateString('thing')},
                {'word': 'level', 'translation': translateString('level')},
                {'word': 'sense', 'translation': translateString('sense')},
                {'word': 'range', 'translation': translateString('range')},
                {'word': 'table', 'translation': translateString('table')},
                {'word': 'value', 'translation': translateString('value')},
                {'word': 'music', 'translation': translateString('music')},
                {'word': 'award', 'translation': translateString('award')},
                {'word': 'space', 'translation': translateString('space')},
                {'word': 'glass', 'translation': translateString('glass')},
                {'word': 'stone', 'translation': translateString('stone')}
            ]
        })))
    pass


def initializeLetterTranslations():
    """Creates a json file called characterConversions.json with the translation
with each letter's translation in morse code"""
    with open('/sd/characterConversions.json', 'w+') as fs:
        fs.write(str(json.dumps({
            'A': '.-',
            'B': '-...',
            'C': '-.-.',
            'D': '-..',
```

```
            'E': '.',
            'F': '..-.',
            'G': '--.',
            'H': '....',
            'I': '..',
            'J': '.---',
            'K': '-.-',
            'L': '.-..',
            'M': '--',
            'N': '-.',
            'O': '---',
            'P': '.--.',
            'Q': '--.-',
            'R': '.-.',
            'S': '...',
            'T': '-',
            'U': '..-',
            'V': '...-',
            'W': '.--',
            'X': '-..-',
            'Y': '-.--',
            'Z': '--..'
        }))))


def translateString(string):
    """Takes a string input and returns the morse code translation as a string
with '.' '-' and '/' symbols"""
    output = ""
    i_end = float(len(string))
    for i in (1 <= i_end) and upRange(1, i_end, 1) or downRange(1, i_end,
1):    # iterates through each character in the string
        if string[int(i - 1)] == ' ':
            output = output +
'/'                                             # for spaces between words
insert a '/' symbol
        else:
            with open('/sd/characterConversions.json', 'r') as
fs:                 # opens the characterConversion.json file
                translations = json.loads((fs.read()))
                output = output + " " + translations[string[int(i - 1)].upper()]
# concatenates the current letter translation to the output string with a

# space separating from the previous letter
```

```
    return output


def broadcastMessage(message):
    """Broadcasts an english string message to the LED and Speaker sensors """
    morseMessage = translateString(message)                                    #
translates the input string into morse code symbols
    j_end = float(len(morseMessage))
    for j in (1 <= j_end) and upRange(1, j_end, 1) or downRange(1, j_end, 1): #
iterates through the morse code string
        if morseMessage[int(j - 1)]=='.':                                      #
for each symbol class their respective functions
            dot()
            sameLetterSpace()                                                  #
call sameLetterSpace() after dot and dash because there is no symbol
        elif morseMessage[int(j - 1)]=='-':                                    #
signifying the space between symbols
            dash()
            sameLetterSpace()
        elif morseMessage[int(j - 1)]==' ':
            letterSpace()
        elif morseMessage[int(j - 1)]=='/':
            slash()


def dot():
    """Outputs the dot symbol on the LED and Speaker sensors"""
    global PWM1, pin0
    for count in range(20):
        PWM1 = machine.PWM(32, freq=12000, duty=50, timer=0)
        wait(0.01)
        PWM1.pause()
        PWM1.freq(12000)
    pin0 = machine.Pin(27, mode=machine.Pin.OUT, pull=machine.Pin.PULL_UP)
    pin0.on()
    wait_ms(100)
    pin0.off()


def dash():
    """Outputs the dash symbol on the LED and Speaker sensors"""
    global PWM1, pin0
    for count in range(50):
        PWM1 = machine.PWM(32, freq=12000, duty=50, timer=0)
        wait(0.01)
```

```
        PWM1.pause()
        PWM1.freq(12000)
    pin0 = machine.Pin(27, mode=machine.Pin.OUT, pull=machine.Pin.PULL_UP)
    pin0.on()
    wait_ms(300)
    pin0.off()


def letterSpace():
    """Waits for the space between letters"""
    wait_ms(1000)


def slash():
    """Waits for the space between words"""
    wait_ms(3000)


def sameLetterSpace():
    """Waits for the space between symbols in the same letter"""
    wait_ms(500)


def upRange(start, stop, step):
    """Auto-generated functions from UI Flow for iterating in a loop"""
    while start <= stop:
        yield start
        start += abs(step)


def downRange(start, stop, step):
    """Auto-generated functions from UI Flow for iterating in a loop"""
    while start >= stop:
        yield start
        start -= abs(step)



# initializeLetterTranslations()
# initializePuzzleWords()
setUpLoginPage()

while not hasLoggedIn:
    wait_ms(2)
```

```
screen.clean_screen()                                    # after logging in
successfully clears the screen of the current UI elements
screen.set_screen_bg_color(0x6998AB)
broadcastLabel = M5Label(                                # creates a label to
show the status of broadcasting messages
    'Waiting for message...',
    x=17,
    y=107,
    color=0xffc900,
    font=FONT_MONT_26,
    parent=None
)
while True:
    broadcastLabel.set_text('Waiting for message...')    # resets the label to
waiting for every every iteration
    broadcastLabel.set_pos(17, 107)
    try:
        req = urequests.request(                         # makes a request to
get all the messages from the messages collection
            method='GET',
            url='http://' + host + ':' + port + '/messages'
        )
        data = json.loads(req.text)                      # loads the result as
a json
        for message in data['_items']:                   # iterates through
each retrieved message
            if not message['sent']:                      # checks if the
current message has not been sent
                broadcastLabel.set_text(message['sent'])
                broadcastLabel.set_text('Broadcasting...') # sets the label to
reflect the broadcasting status
                broadcastLabel.set_pos(64, 107)
                for i in range(0, int(message['numberOfTimes'])):
                    broadcastMessage(message['message'])   # broadcasts the
current message the number of times specified in the application
                message['sent'] = True                   # sets the current
message as being sent
                req = urequests.request(                 # updates the message
in the database
                    method='PATCH',
                    url='http://' + host + ':' + port + '/messages',
                    json=message
                )
    except:
        broadcastLabel.set_text('Error: Failed to Connect')
```

```
        broadcastLabel.set_pos(8, 107)
        wait(1)
    wait_ms(2)
```

23.2 FlowPuzzle.py

```
"""
Project: Team 3 Capstone
Purpose Details: Implementation of the Flow Puzzle
Course: CMPSC 488/ IST 440W
Author: Bradley Whitebread
Date Developed: 3/23/2022
Last Date Changed: 4/3/2022
Revision: 4
"""

from m5stack import *
from m5stack_ui import *
from uiflow import *
from m5stack import touch
from m5mqtt import M5mqtt
import wifiCfg


screen = M5Screen()
screen.clean_screen()
screen.set_screen_bg_color(0x6998AB)

wifiCfg.doConnect('Pixel_5294', 'bradley239')
m5mqtt = M5mqtt('adafruit', 'io.adafruit.com', 1883, 'bew5294',
'aio_pLPP82smfxwykOdHBbsug5RXEBM5', 300)
m5mqtt.start()


# Variables
host = '192.168.1.196'
port = '49153'

puzzleStartTime = None
puzzleEndTime = None

lastPressed = 'black' # lastPressed is used to store the string of the last
pressed root node
notReleased = True
color = [0, 0, 0]
```

```
# added None padding to all 2D arrays for easier index accessing
# declares a 2D array that is the solution to the flow puzzle
solution = [
    [None, None, None, None, None, None],
    [None, 'indigo', 'indigo', 'indigo', 'red',    'red',    'green'],
    [None, 'indigo', 'orange', 'indigo', 'yellow', 'red',    'green'],
    [None, 'indigo', 'orange', 'indigo', 'yellow', 'red',    'green'],
    [None, 'indigo', 'orange', 'indigo', 'yellow', 'blue',   'orange'],
    [None, 'indigo', 'orange', 'blue',   'blue',   'blue',   'orange'],
    [None, 'indigo', 'orange', 'orange', 'orange', 'orange', 'orange']
]

# declares a 2D array that tracks the current position of the flow puzzle
btnTagMatrix = [
    [None, None, None, None, None, None],
    [None, 'black',  'black',  'black',  'red',    'black', 'green'],
    [None, 'black',  'orange', 'black',  'yellow', 'black', 'black'],
    [None, 'black',  'black',  'black',  'black',  'red',   'green'],
    [None, 'black',  'black',  'indigo', 'yellow', 'blue',  'orange'],
    [None, 'black',  'black',  'blue',   'black',  'black', 'black'],
    [None, 'indigo', 'black',  'black',  'black',  'black', 'black']
]

# declares a 2D array of the buttons in the flow puzzle
btnMatrix = [
    [None, None, None, None, None, None],
    [
        None,
        M5Btn(text='o', x=40, y=0, w=40, h=40, bg_c=0x000000,
              text_c=0x000000, font=FONT_MONT_14, parent=None),
        M5Btn(text='o', x=80, y=0, w=40, h=40, bg_c=0x000000,
              text_c=0x000000, font=FONT_MONT_14, parent=None),
        M5Btn(text='o', x=120, y=0, w=40, h=40, bg_c=0x000000,
              text_c=0x000000, font=FONT_MONT_14, parent=None),
        M5Btn(text='o', x=160, y=0, w=40, h=40, bg_c=0xff0000,
              text_c=0x000000, font=FONT_MONT_14, parent=None),
        M5Btn(text='o', x=200, y=0, w=40, h=40, bg_c=0x000000,
              text_c=0x000000, font=FONT_MONT_14, parent=None),
        M5Btn(text='o', x=240, y=0, w=40, h=40, bg_c=0x006600,
              text_c=0x000000, font=FONT_MONT_14, parent=None)
    ],
    [
        None,
        M5Btn(text='o', x=40, y=40, w=40, h=40, bg_c=0x000000,
              text_c=0x000000, font=FONT_MONT_14, parent=None),
```

```
        M5Btn(text='o', x=80, y=40, w=40, h=40, bg_c=0xff9900,
              text_c=0x000000, font=FONT_MONT_14, parent=None),
        M5Btn(text='o', x=120, y=40, w=40, h=40, bg_c=0x000000,
              text_c=0x000000, font=FONT_MONT_14, parent=None),
        M5Btn(text='o', x=160, y=40, w=40, h=40, bg_c=0xffff00,
              text_c=0x000000, font=FONT_MONT_14, parent=None),
        M5Btn(text='o', x=200, y=40, w=40, h=40, bg_c=0x000000,
              text_c=0x000000, font=FONT_MONT_14, parent=None),
        M5Btn(text='o', x=240, y=40, w=40, h=40, bg_c=0x000000,
              text_c=0x000000, font=FONT_MONT_14, parent=None)
    ],
    [
        None,
        M5Btn(text='o', x=40, y=80, w=40, h=40, bg_c=0x000000,
              text_c=0x000000, font=FONT_MONT_14, parent=None),
        M5Btn(text='o', x=80, y=80, w=40, h=40, bg_c=0x000000,
              text_c=0x000000, font=FONT_MONT_14, parent=None),
        M5Btn(text='o', x=120, y=80, w=40, h=40, bg_c=0x000000,
              text_c=0x000000, font=FONT_MONT_14, parent=None),
        M5Btn(text='o', x=160, y=80, w=40, h=40, bg_c=0x000000,
              text_c=0x000000, font=FONT_MONT_14, parent=None),
        M5Btn(text='o', x=200, y=80, w=40, h=40, bg_c=0xff0000,
              text_c=0x000000, font=FONT_MONT_14, parent=None),
        M5Btn(text='o', x=240, y=80, w=40, h=40, bg_c=0x006600,
              text_c=0x000000, font=FONT_MONT_14, parent=None)
    ],
    [
        None,
        M5Btn(text='o', x=40, y=120, w=40, h=40, bg_c=0x000000,
              text_c=0x000000, font=FONT_MONT_14, parent=None),
        M5Btn(text='o', x=80, y=120, w=40, h=40, bg_c=0x000000,
              text_c=0x000000, font=FONT_MONT_14, parent=None),
        M5Btn(text='o', x=120, y=120, w=40, h=40, bg_c=0x3333ff,
              text_c=0x000000, font=FONT_MONT_14, parent=None),
        M5Btn(text='o', x=160, y=120, w=40, h=40, bg_c=0xffff00,
              text_c=0x000000, font=FONT_MONT_14, parent=None),
        M5Btn(text='o', x=200, y=120, w=40, h=40, bg_c=0x33ffff,
              text_c=0x000000, font=FONT_MONT_14, parent=None),
        M5Btn(text='o', x=240, y=120, w=40, h=40, bg_c=0xff9900,
              text_c=0x000000, font=FONT_MONT_14, parent=None)
    ],
    [
        None,
        M5Btn(text='o', x=40, y=160, w=40, h=40, bg_c=0x000000,
              text_c=0x000000, font=FONT_MONT_14, parent=None),
```

```
        M5Btn(text='o', x=80, y=160, w=40, h=40, bg_c=0x000000,
              text_c=0x000000, font=FONT_MONT_14, parent=None),
        M5Btn(text='o', x=120, y=160, w=40, h=40, bg_c=0x33ffff,
              text_c=0x000000, font=FONT_MONT_14, parent=None),
        M5Btn(text='o', x=160, y=160, w=40, h=40, bg_c=0x000000,
              text_c=0x000000, font=FONT_MONT_14, parent=None),
        M5Btn(text='o', x=200, y=160, w=40, h=40, bg_c=0x000000,
              text_c=0x000000, font=FONT_MONT_14, parent=None),
        M5Btn(text='o', x=240, y=160, w=40, h=40, bg_c=0x000000,
              text_c=0x000000, font=FONT_MONT_14, parent=None)
    ],
    [
        None,
        M5Btn(text='o', x=40, y=200, w=40, h=40, bg_c=0x3333ff,
              text_c=0x000000, font=FONT_MONT_14, parent=None),
        M5Btn(text='o', x=80, y=200, w=40, h=40, bg_c=0x000000,
              text_c=0x000000, font=FONT_MONT_14, parent=None),
        M5Btn(text='o', x=120, y=200, w=40, h=40, bg_c=0x000000,
              text_c=0x000000, font=FONT_MONT_14, parent=None),
        M5Btn(text='o', x=160, y=200, w=40, h=40, bg_c=0x000000,
              text_c=0x000000, font=FONT_MONT_14, parent=None),
        M5Btn(text='o', x=200, y=200, w=40, h=40, bg_c=0x000000,
              text_c=0x000000, font=FONT_MONT_14, parent=None),
        M5Btn(text='o', x=240, y=200, w=40, h=40, bg_c=0x000000,
              text_c=0x000000, font=FONT_MONT_14, parent=None)
    ]
]

for row in range(1, len(btnMatrix)):              # hides all the buttons in the
btn matrix
    for col in range(1, len(btnMatrix[row])):
        btnMatrix[row][col].set_hidden(True)


# Functions
def setUpFlowPuzzle():
    """Hides the login page and displays all the buttons in the flow puzzle"""
    global btnMatrix, puzzleStartTime
    hideLoginPage()                               # hides login page
    for row in range(1, len(btnMatrix)):          # shows all the buttons in the
btn matrix
        for col in range(1, len(btnMatrix[row])):
            btnMatrix[row][col].set_hidden(False)
    puzzleStartTime = time.ticks_ms()
```

```
def checkFlowPuzzle():
    """Checks if every button in the puzzle is correctly set. There is only one
correct configuration for the puzzle.
       If the puzzle is correct a log is posted to the database saying the Flow
Puzzle was solved"""
    global color, notReleased, lastPressed, btnTagMatrix, puzzleStartTime,
puzzleEndTime, hasLoggedIn
    if btnTagMatrix == solution:                    # checks if btn tag matrix
is equal to the solution matrix
        puzzleEndTime = time.ticks_ms()             # finished tracking how
long it takes to complete the puzzle
        duration = puzzleEndTime - puzzleStartTime  # calculates puzzle
duration
        m5mqtt.publish('bew5294/feeds/capstone-morse-communicator',
'maintenance')
        hasLoggedIn = True
        try:
            req = urequests.request(                # posts json to log
database with the puzzle name, status, and duration
                method='POST',
                url='http://' + host + ':' + port + '/logs',
                json={
                    'app':'Flow Puzzle',
                    'status':'completed',
                    'duration':duration
                }
            )
        except:
            pass
    pass


def changeBtnColor(row, col):
    """Changes the color of the button at the inputted coordinates"""
    global color, notReleased, lastPressed, btnMatrix, btnTagMatrix
    if notReleased:                                 # checks that the
screen has been continuously pressed since touching a root node
        btnMatrix[row][col].set_bg_color(           # sets the button
color to the color of the last pressed root node
            (color[0] << 16) | (color[1] << 8) | color[2])
        btnMatrix[row][col].set_btn_text_color(     # sets the button text
color to the color of the last pressed root node
            (color[0] << 16) | (color[1] << 8) | color[2])
        btnTagMatrix[row][col] = lastPressed        # sets the button tag
to the last pressed root node color
```

```
    pass


def setLastPressed(colorName):
    """Sets the current color and lastPressed variable to that of the inputted
string"""
    global color, lastPressed, notReleased

    notReleased = True

    if colorName == 'red':
        color = [255, 0, 0]
        lastPressed = 'red'
        pass

    elif colorName == 'orange':
        color = [255, 153, 0]
        lastPressed = 'orange'
        pass

    elif colorName == 'yellow':
        color = [255, 255, 0]
        lastPressed = 'yellow'
        pass

    elif colorName == 'green':
        color = [0, 102, 0]
        lastPressed = 'green'
        pass

    elif colorName == 'blue':
        color = [51, 255, 255]
        lastPressed = 'blue'
        pass

    elif colorName == 'indigo':
        color = [51, 51, 255]
        lastPressed = 'indigo'
        pass
    pass

# root buttons (root refers to one end of a line that cannot be changed)
def redRootPressed():
    """Sets last pressed to 'red'"""
    setLastPressed('red')
```

```
    pass
btnMatrix[1][4].pressed(redRootPressed)
btnMatrix[3][5].pressed(redRootPressed)


def orangeRootPressed():
    """Sets last pressed to 'orange'"""
    setLastPressed('orange')
    pass
btnMatrix[2][2].pressed(orangeRootPressed)
btnMatrix[4][6].pressed(orangeRootPressed)


def yellowRootPressed():
    """Sets last pressed to 'yellow'"""
    setLastPressed('yellow')
    pass
btnMatrix[2][4].pressed(yellowRootPressed)
btnMatrix[4][4].pressed(yellowRootPressed)


def greenRootPressed():
    """Sets last pressed to 'green'"""
    setLastPressed('green')
    pass
btnMatrix[1][6].pressed(greenRootPressed)
btnMatrix[3][6].pressed(greenRootPressed)


def blueRootPressed():
    """Sets last pressed to 'blue'"""
    setLastPressed('blue')
    pass
btnMatrix[4][5].pressed(blueRootPressed)
btnMatrix[5][3].pressed(blueRootPressed)


def indigoRootPressed():
    """Sets last pressed to 'indigo'"""
    setLastPressed('indigo')
    pass
btnMatrix[4][3].pressed(indigoRootPressed)
btnMatrix[6][1].pressed(indigoRootPressed)


# non-root buttons (the color of these buttons can be changed)
```

```
def btn11Pressed():
    changeBtnColor(1, 1)
    pass
btnMatrix[1][1].pressed(btn11Pressed)


def btn12Pressed():
    changeBtnColor(1, 2)
    pass
btnMatrix[1][2].pressed(btn12Pressed)


def btn13Pressed():
    changeBtnColor(1, 3)
    pass
btnMatrix[1][3].pressed(btn13Pressed)


def btn15Pressed():
    changeBtnColor(1, 5)
    pass
btnMatrix[1][5].pressed(btn15Pressed)


def btn21Pressed():
    changeBtnColor(2, 1)
    pass
btnMatrix[2][1].pressed(btn21Pressed)


def btn23Pressed():
    changeBtnColor(2, 3)
    pass
btnMatrix[2][3].pressed(btn23Pressed)


def btn25Pressed():
    changeBtnColor(2, 5)
    pass
btnMatrix[2][5].pressed(btn25Pressed)


def btn26Pressed():
    changeBtnColor(2, 6)
    pass
btnMatrix[2][6].pressed(btn26Pressed)
```

```
def btn31Pressed():
    changeBtnColor(3, 1)
    pass
btnMatrix[3][1].pressed(btn31Pressed)

def btn32Pressed():
    changeBtnColor(3, 2)
    pass
btnMatrix[3][2].pressed(btn32Pressed)

def btn33Pressed():
    changeBtnColor(3, 3)
    pass
btnMatrix[3][3].pressed(btn33Pressed)

def btn34Pressed():
    changeBtnColor(3, 4)
    pass
btnMatrix[3][4].pressed(btn34Pressed)

def btn41Pressed():
    changeBtnColor(4, 1)
    pass
btnMatrix[4][1].pressed(btn41Pressed)

def btn42Pressed():
    changeBtnColor(4, 2)
    pass
btnMatrix[4][2].pressed(btn42Pressed)

def btn51Pressed():
    changeBtnColor(5, 1)
    pass
btnMatrix[5][1].pressed(btn51Pressed)

def btn52Pressed():
    changeBtnColor(5, 2)
```

```
    pass
btnMatrix[5][2].pressed(btn52Pressed)


def btn54Pressed():
    changeBtnColor(5, 4)
    pass
btnMatrix[5][4].pressed(btn54Pressed)


def btn55Pressed():
    changeBtnColor(5, 5)
    pass
btnMatrix[5][5].pressed(btn55Pressed)


def btn56Pressed():
    changeBtnColor(5, 6)
    pass
btnMatrix[5][6].pressed(btn56Pressed)


def btn62Pressed():
    changeBtnColor(6, 2)
    pass
btnMatrix[6][2].pressed(btn62Pressed)


def btn63Pressed():
    changeBtnColor(6, 3)
    pass
btnMatrix[6][3].pressed(btn63Pressed)


def btn64Pressed():
    changeBtnColor(6, 4)
    pass
btnMatrix[6][4].pressed(btn64Pressed)


def btn65Pressed():
    changeBtnColor(6, 5)
    pass
btnMatrix[6][5].pressed(btn65Pressed)
```

```
def btn66Pressed():
    changeBtnColor(6, 6)
    pass
btnMatrix[6][6].pressed(btn66Pressed)


def btnReleased():
    """When a button is released the status of notReleased is updated and the
puzzle is checked if it is solved"""
    global notReleased
    notReleased = touch.status() # updates the status of notReleased (False if
the screen has been released)
    checkFlowPuzzle()            # checks if the puzzle has been solved
    pass
btnMatrix[1][1].released(btnReleased)
btnMatrix[1][2].released(btnReleased)
btnMatrix[1][3].released(btnReleased)
btnMatrix[1][4].released(btnReleased)
btnMatrix[1][5].released(btnReleased)
btnMatrix[1][6].released(btnReleased)

btnMatrix[2][1].released(btnReleased)
btnMatrix[2][2].released(btnReleased)
btnMatrix[2][3].released(btnReleased)
btnMatrix[2][4].released(btnReleased)
btnMatrix[2][5].released(btnReleased)
btnMatrix[2][6].released(btnReleased)

btnMatrix[3][1].released(btnReleased)
btnMatrix[3][2].released(btnReleased)
btnMatrix[3][3].released(btnReleased)
btnMatrix[3][4].released(btnReleased)
btnMatrix[3][5].released(btnReleased)
btnMatrix[3][6].released(btnReleased)

btnMatrix[4][1].released(btnReleased)
btnMatrix[4][2].released(btnReleased)
btnMatrix[4][3].released(btnReleased)
btnMatrix[4][4].released(btnReleased)
btnMatrix[4][5].released(btnReleased)
btnMatrix[4][6].released(btnReleased)

btnMatrix[5][1].released(btnReleased)
btnMatrix[5][2].released(btnReleased)
btnMatrix[5][3].released(btnReleased)
```

```
btnMatrix[5][4].released(btnReleased)
btnMatrix[5][5].released(btnReleased)
btnMatrix[5][6].released(btnReleased)

btnMatrix[6][1].released(btnReleased)
btnMatrix[6][2].released(btnReleased)
btnMatrix[6][3].released(btnReleased)
btnMatrix[6][4].released(btnReleased)
btnMatrix[6][5].released(btnReleased)
btnMatrix[6][6].released(btnReleased)


setUpFlowPuzzle()
```

23.3 MorsePuzzle.py

```
"""
Project: Team 3 Capstone
Purpose Details: Implementation of the Morse Code Puzzle
Course: CMPSC 488/ IST 440W
Author: Bradley Whitebread
Date Developed: 3/23/2022
Last Date Changed: 4/3/2022
Revision: 2
"""

from m5stack import *
from m5stack_ui import *
from uiflow import *
from m5mqtt import M5mqtt
import wifiCfg
import json
import time
import random


screen = M5Screen()
screen.clean_screen()
screen.set_screen_bg_color(0x6998AB)

wifiCfg.doConnect('Pixel_5294', 'bradley239')
m5mqtt = M5mqtt('adafruit', 'io.adafruit.com', 1883, 'bew5294',
'aio_pLPP82smfxwykOdHBbsug5RXEBM5', 300)
m5mqtt.start()
```

```
# Variables
host = '192.168.1.196'
port = '49153'

puzzleStartTime = None
puzzleEndTime = None

morseAnswer = None
morseOptions = []

morseLabel = M5Label('Text', x=0, y=77, color=0x000,
                     font=FONT_MONT_14, parent=None)
morseLabel.set_hidden(True)

morseOptionBtns = [
    M5Btn(text='Button', x=15,  y=15, w=290, h=60, bg_c=0xFFFFFF,
          text_c=0x000000, font=FONT_MONT_44, parent=None),
    M5Btn(text='Button', x=15,  y=90, w=290, h=60, bg_c=0xFFFFFF,
          text_c=0x000000, font=FONT_MONT_44, parent=None),
    M5Btn(text='Button', x=15, y=165, w=290, h=60, bg_c=0xFFFFFF,
          text_c=0x000000, font=FONT_MONT_44, parent=None)
]
morseOptionBtns[0].set_hidden(True)
morseOptionBtns[1].set_hidden(True)
morseOptionBtns[2].set_hidden(True)

# Functions
def setUpMorsePuzzle():
    """Sets up and runs the Morse Puzzle. Hides the login page then picks and
broadcasts a word in Morse code.
       The user must decode and pick the word's english translation from three
options"""
    global morseOptions, morseAnswer, puzzleStartTime
    hideLoginPage()                                          # hides the
login page
    morseOptions = pickOptions()                             # picks
three options from the puzzle words json file
    morseAnswer = morseOptions[int(random.randint(1, 3) - 1)]['word'] # picks the
answer from the three options
    puzzleStartTime = time.ticks_ms()                        # starts
tracking how long it takes to complete the puzzle
    broadcastMessage(morseAnswer)                            # outputs
the answer in morse code
    displayOptions(morseOptions)                             # displays
the more option buttons
```

```
def pickOptions():
    """Picks three unique words from puzzleWords.json"""
    with open('/sd/puzzleWords.json', 'r') as fs:                # opens
puzzleWords.json
        allWords = json.loads((fs.read()))                       # loads json
file into allWords dictionary

    wordList = allWords['words']                                 # takes all
entries in the 'words' key from puzzleWords.json
    randIndex1 = random.randint(1, len(wordList))
    randIndex2 = random.randint(1, len(wordList))
    while randIndex1 == randIndex2:
        randIndex2 = random.randint(1, len(wordList))

    randIndex3 = random.randint(1, len(wordList))
    while randIndex3 == randIndex1 or randIndex3 == randIndex2:
        randIndex3 = random.randint(1, len(wordList))

    morseOptions = [                                             # fills
morseOptions array with the three unique words
        wordList[int(randIndex1 - 1)],
        wordList[int(randIndex2 - 1)],
        wordList[int(randIndex3 - 1)]
    ]
    return morseOptions


def displayOptions(wordOptions):
    """Displays all the morseOption buttons"""
    global morseOptionBtns
    morseOptionBtns[0].set_btn_text((wordOptions[0]['word'])) # sets the button
text to each word
    morseOptionBtns[1].set_btn_text((wordOptions[1]['word']))
    morseOptionBtns[2].set_btn_text((wordOptions[2]['word']))
    morseOptionBtns[0].set_hidden(False)                        # displays all more
options buttons
    morseOptionBtns[1].set_hidden(False)
    morseOptionBtns[2].set_hidden(False)


def morse1BtnPressed():
    """Checks if the first word is the correct translation. If it is a log is
posted to the database saying the Morse Puzzle was solved.
```

```
       If not the Morse Puzzle is reset for another try with a new word"""
    global morseOptions, puzzleStartTime, puzzleEndTime, host, port, hasLoggedIn
    if morseAnswer == (morseOptions[0]['word']):
        screen.set_screen_bg_color(0x99ff99)
        puzzleEndTime = time.ticks_ms()                    # finished tracking
how long it takes to complete the puzzle
        duration = puzzleEndTime - puzzleStartTime         # calculates puzzle
duration
        m5mqtt.publish('bew5294/feeds/capstone-morse-communicator', 'admin')
        hasLoggedIn = True
        try:
            req = urequests.request(
                method='POST',
                url='http://' + host + ':' + port + '/logs',
                json={
                    'app': 'Morse Puzzle',
                    'status': 'success',
                    'duration': duration
                }
            )
        except:
            pass
    else:
        screen.set_screen_bg_color(0xffcccc)               # sets the screen
red if tone 3 is the incorrect answer
        wait(1)
        setUpMorsePuzzle()                                 # also resets the
puzzle to retry with new words
    pass
morseOptionBtns[0].pressed(morse1BtnPressed)


def morse2BtnPressed():
    """Checks if the second word is the correct translation. If it is a log is
posted to the database saying the Morse Puzzle was solved.
       If not the Morse Puzzle is reset for another try with a new word"""
    global morseOptions, puzzleStartTime, puzzleEndTime, host, port, hasLoggedIn
    if morseAnswer == (morseOptions[1]['word']):
        screen.set_screen_bg_color(0x99ff99)
        puzzleEndTime = time.ticks_ms()                    # finished tracking
how long it takes to complete the puzzle
        duration = puzzleEndTime - puzzleStartTime         # calculates puzzle
duration
        m5mqtt.publish('bew5294/feeds/capstone-morse-communicator', 'admin')
        hasLoggedIn = True
```

```
        try:
            req = urequests.request(
                method='POST',
                url='http://' + host + ':' + port + '/logs',
                json={
                    'app': 'Sound Puzzle',
                    'status': 'success',
                    'duration': duration
                }
            )
        except:
            pass
    else:
        screen.set_screen_bg_color(0xffcccc)                # sets the screen
red if tone 3 is the incorrect answer
        wait(1)
        setUpMorsePuzzle()                                  # also resets the
puzzle to retry with new words
    pass
morseOptionBtns[1].pressed(morse2BtnPressed)


def morse3BtnPressed():
    """Checks if the third word is the correct translation. If it is a log is
posted to the database saying the Morse Puzzle was solved.
       If not the Morse Puzzle is reset for another try with a new word"""
    global morseOptions, puzzleStartTime, puzzleEndTime, host, port, hasLoggedIn
    if morseAnswer == (morseOptions[2]['word']):
        screen.set_screen_bg_color(0x99ff99)
        puzzleEndTime = time.ticks_ms()                     # finished tracking
how long it takes to complete the puzzle
        duration = puzzleEndTime - puzzleStartTime          # calculates puzzle
duration
        m5mqtt.publish('bew5294/feeds/capstone-morse-communicator', 'admin')
        hasLoggedIn = True
        try:
            req = urequests.request(
                method='POST',
                url='http://' + host + ':' + port + '/logs',
                json={
                    'app': 'Sound Puzzle',
                    'status': 'success',
                    'duration': duration
                }
            )
```

```
        except:
            pass
    else:
        screen.set_screen_bg_color(0xffcccc)                # sets the screen
red if tone 3 is the incorrect answer
        wait(1)
        setUpMorsePuzzle()                                  # also resets the
puzzle to retry with new words
    pass
morseOptionBtns[2].pressed(morse3BtnPressed)


def initializePuzzleWords():
    """Creates a json file called puzzleWords.json with 20 puzzle words to use in
the Morse Puzzle"""
    with open('/sd/puzzleWords.json', 'w+') as fs:
        fs.write(str(json.dumps({
            'words': [                                              #
creates a list of dictionaries
                {'word': 'world', 'translation': translateString('world')}, #
creates a dictionary of words with a word key and a key for the morse translation
                {'word': 'house', 'translation': translateString('house')},
                {'word': 'place', 'translation': translateString('place')},
                {'word': 'group', 'translation': translateString('group')},
                {'word': 'flame', 'translation': translateString('flame')},
                {'word': 'clock', 'translation': translateString('clock')},
                {'word': 'point', 'translation': translateString('point')},
                {'word': 'night', 'translation': translateString('night')},
                {'word': 'water', 'translation': translateString('water')},
                {'word': 'thing', 'translation': translateString('thing')},
                {'word': 'level', 'translation': translateString('level')},
                {'word': 'sense', 'translation': translateString('sense')},
                {'word': 'range', 'translation': translateString('range')},
                {'word': 'table', 'translation': translateString('table')},
                {'word': 'value', 'translation': translateString('value')},
                {'word': 'music', 'translation': translateString('music')},
                {'word': 'award', 'translation': translateString('award')},
                {'word': 'space', 'translation': translateString('space')},
                {'word': 'glass', 'translation': translateString('glass')},
                {'word': 'stone', 'translation': translateString('stone')}
            ]
        })))
    pass
```

```
def initializeLetterTranslations():
    """Creates a json file called characterConversions.json with the translation
with each letter's translation in morse code"""
    with open('/sd/characterConversions.json', 'w+') as fs:
        fs.write(str(json.dumps(({
            'A': '.-',
            'B': '-...',
            'C': '-.-.',
            'D': '-..',
            'E': '.',
            'F': '..-.',
            'G': '--.',
            'H': '....',
            'I': '..',
            'J': '.---',
            'K': '-.-',
            'L': '.-..',
            'M': '--',
            'N': '-.',
            'O': '---',
            'P': '.--.',
            'Q': '--.-',
            'R': '.-.',
            'S': '...',
            'T': '-',
            'U': '..-',
            'V': '...-',
            'W': '.--',
            'X': '-..-',
            'Y': '-.--',
            'Z': '--..'
        }))))


def translateString(string):
    """Takes a string input and returns the morse code translation as a string
with '.' '-' and '/' symbols"""
    output = ""
    i_end = float(len(string))
    for i in (1 <= i_end) and upRange(1, i_end, 1) or downRange(1, i_end,
1):    # iterates through each character in the string
        if string[int(i - 1)] == ' ':
            output = output +
'/'                                          # for spaces between words
insert a '/' symbol
```

```
        else:
            with open('/sd/characterConversions.json', 'r') as
fs:                 # opens the characterConversion.json file
                translations = json.loads((fs.read()))
                output = output + " " + translations[string[int(i - 1)].upper()]
# concatenates the current letter translation to the output string with a

# space separating from the previous letter

    return output


def broadcastMessage(message):
    """Broadcasts an english string message to the LED and Speaker sensors """
    morseMessage = translateString(message)                                    #
translates the input string into morse code symbols
    j_end = float(len(morseMessage))
    for j in (1 <= j_end) and upRange(1, j_end, 1) or downRange(1, j_end, 1): #
iterates through the morse code string
        if morseMessage[int(j - 1)]=='.':                                      #
for each symbol class their respective functions
            dot()
            sameLetterSpace()                                                  #
call sameLetterSpace() after dot and dash because there is no symbol
        elif morseMessage[int(j - 1)]=='-':                                    #
signifying the space between symbols
            dash()
            sameLetterSpace()
        elif morseMessage[int(j - 1)]==' ':
            letterSpace()
        elif morseMessage[int(j - 1)]=='/':
            slash()


def dot():
    """Outputs the dot symbol on the LED and Speaker sensors"""
    global PWM1, pin0
    for count in range(20):
        PWM1 = machine.PWM(32, freq=12000, duty=50, timer=0)
        wait(0.01)
        PWM1.pause()
        PWM1.freq(12000)
    pin0 = machine.Pin(27, mode=machine.Pin.OUT, pull=machine.Pin.PULL_UP)
    pin0.on()
    wait_ms(100)
```

```
    pin0.off()


def dash():
    """Outputs the dash symbol on the LED and Speaker sensors"""
    global PWM1, pin0
    for count in range(50):
        PWM1 = machine.PWM(32, freq=12000, duty=50, timer=0)
        wait(0.01)
        PWM1.pause()
        PWM1.freq(12000)
    pin0 = machine.Pin(27, mode=machine.Pin.OUT, pull=machine.Pin.PULL_UP)
    pin0.on()
    wait_ms(300)
    pin0.off()


def letterSpace():
    """Waits for the space between letters"""
    wait_ms(1000)


def slash():
    """Waits for the space between words"""
    wait_ms(3000)


def sameLetterSpace():
    """Waits for the space between symbols in the same letter"""
    wait_ms(500)


def upRange(start, stop, step):
    """Auto-generated functions from UI Flow for iterating in a loop"""
    while start <= stop:
        yield start
        start += abs(step)


def downRange(start, stop, step):
    """Auto-generated functions from UI Flow for iterating in a loop"""
    while start >= stop:
        yield start
        start -= abs(step)

setUpMorsePuzzle()
```

23.4 BroadcastMessage.py

```
"""
Project: Team 3 Capstone
Purpose Details: Broadcast Message Method for M5Stack
Course: CMPSC 488/ IST 440W
Author: Md Mollah
Date Developed: 3/27/2022
Last Date Changed: 4/8/2022
Revision: 2
"""

from m5stack import *
from m5stack_ui import *
from uiflow import *
import machine
import time
import json


screen = M5Screen()
screen.clean_screen()
screen.set_screen_bg_color(0xFFFFFF)


string = None
message = None
output = None
morseMessage = None
i = None
j = None
translations = None



option1_btn = M5Btn(text='Button', x=125, y=21, w=70, h=30, bg_c=0xFFFFFF, text_c=0x000000, font=FONT_MONT_14, parent=None)
touch_button1 = M5Btn(text='Button', x=124, y=71, w=70, h=30, bg_c=0xFFFFFF, text_c=0x000000, font=FONT_MONT_14, parent=None)
touch_button2 = M5Btn(text='Button', x=126, y=123, w=70, h=30, bg_c=0xFFFFFF, text_c=0x000000, font=FONT_MONT_14, parent=None)
label0 = M5Label('Text', x=6, y=13, color=0x000, font=FONT_MONT_14, parent=None)
label1 = M5Label('Text', x=8, y=73, color=0x000, font=FONT_MONT_14, parent=None)
label2 = M5Label('Text', x=7, y=135, color=0x000, font=FONT_MONT_14, parent=None)
label3 = M5Label('Text', x=148, y=189, color=0x000, font=FONT_MONT_14, parent=None)
```

```
""" Defining the translation of morse code letter by letter"""
def initializeLetterTranslations():
  global string, message, output, morseMessage, i, j, translations, pin0
  with open('/sd/' + str('letters.json'), 'w+') as fs:
    fs.write(str(json.dumps(({
      'A':'.-',
      'B':'-...',
      'C':'-.-.',
      'D':'-..',
      'E':'.',
      'F':'..-.',
      'G':'--.',
      'H':'....',
      'I':'..',
      'J':'.---',
      'K':'-.-',
      'L':'.-..',
      'M':'--',
      'N':'-.',
      'O':'---',
      'P':'.--.',
      'Q':'--.-',
      'R':'.-.',
      'S':'...',
      'T':'-',
      'U':'..-',
      'V':'...-',
      'W':'.--',
      'X':'-..-',
      'Y':'-.--',
      'Z':'--..'
    })))))

def upRange(start, stop, step):
  while start <= stop:
    yield start
    start += abs(step)

def downRange(start, stop, step):
  while start >= stop:
    yield start
    start -= abs(step)
```

```
"""Taking in a string and translating it"""
def translateString(string):
  global message, output, morseMessage, i, j, translations, pin0
  output = ""
  i_end = float(len(string))
  for i in (1 <= i_end) and upRange(1, i_end, 1) or downRange(1, i_end, 1):
    if string[int(i - 1)] == ' ':
      output = (str(output) + str('/'))
    else:
      with open('/sd/letters.json', 'r') as fs:
        translations = json.loads((fs.read()))
        output = (str(output) + str(((str(' ') + str((translations[string[int(i - 1)]]))))))
  return output

"""Outputting a dot as sound and light"""
def dot():
  global string, message, output, morseMessage, i, j, translations, pin0
  pin0 = machine.Pin(27, mode=machine.Pin.OUT, pull=machine.Pin.PULL_UP)
  pin0.on()
  wait_ms(500)
  pin0.off()

"""Outputting a dash as sound and light"""
def dash():
  global string, message, output, morseMessage, i, j, translations, pin0
  pin0 = machine.Pin(27, mode=machine.Pin.OUT, pull=machine.Pin.PULL_UP)
  pin0.on()
  wait_ms(1500)
  pin0.off()

"""Outputting a space in the same letter"""
def sameLetterSpace():
  global string, message, output, morseMessage, i, j, translations, pin0
  wait_ms(500)

"""Outputting a space between different letters"""
def letterSpace():
  global string, message, output, morseMessage, i, j, translations, pin0
  wait_ms(1000)

"""Outputting a slash between different words"""
def slash():
  global string, message, output, morseMessage, i, j, translations, pin0
  wait_ms(3000)
```

```
"""Broadcasting the message as morse code"""
def broadcastMessage(message):
  global string, output, morseMessage, i, j, translations, pin0
  morseMessage = translateString(message)
  j_end = float(len(morseMessage))
  for j in (1 <= j_end) and upRange(1, j_end, 1) or downRange(1, j_end, 1):
    if morseMessage[int(j - 1)]=='.':
      dot()
      sameLetterSpace()
    elif morseMessage[int(j - 1)]=='-':
      dash()
      sameLetterSpace()
    elif morseMessage[int(j - 1)]==' ':
      letterSpace()
    elif morseMessage[int(j - 1)]=='/':
      slash()
    else:
      pass
```

23.5 LoginPage.py

```
"""
Project: Team 3 Capstone
Purpose Details: Login Page on the M5 Stack
Course: CMPSC 488/ IST 440W
Author: Bradley Whitebread/Md Mollah
Date Developed: 3/23/2022
Last Date Changed: 4/8/2022
Revision: 2
"""

from m5stack import *
from m5stack_ui import *
from uiflow import *

screen = M5Screen()
screen.clean_screen()
screen.set_screen_bg_color(0x6998ab)


"""Variables"""
roleSelected = False

"""Intializes UI elements for Login Page"""
pickRoleLabel1 = M5Label('Pick Your', x=75, y=29,
                         color=0xffc900, font=FONT_MONT_36, parent=None)
pickRoleLabel2 = M5Label('Role to Login', x=40, y=73,
                         color=0xffc900, font=FONT_MONT_36, parent=None)

defaultLabel = M5Label('Default', x=30, y=172, color=0x000,
                       font=FONT_MONT_14, parent=None)
maintLabel = M5Label('Maintenance', x=113, y=172,
                     color=0x000, font=FONT_MONT_14, parent=None)
adminLabel = M5Label('Admin', x=240, y=172, color=0x000,
                     font=FONT_MONT_14, parent=None)

downArrowImage1 = M5Img("res/DownArrow.png", x=45, y=195, parent=None)
downArrowImage2 = M5Img("res/DownArrow.png", x=150, y=195, parent=None)
downArrowImage3 = M5Img("res/DownArrow.png", x=255, y=195, parent=None)

"""Hides all Login Page UI elements"""
pickRoleLabel1.set_hidden(True)
pickRoleLabel2.set_hidden(True)

defaultLabel.set_hidden(True)
maintLabel.set_hidden(True)
adminLabel.set_hidden(True)

downArrowImage1.set_hidden(True)
downArrowImage2.set_hidden(True)
downArrowImage3.set_hidden(True)
```

```
def setUpLoginPage():
    """Displays of UI elements in the Login Page"""
    pickRoleLabel1.set_hidden(False)
    pickRoleLabel2.set_hidden(False)

    defaultLabel.set_hidden(False)
    maintLabel.set_hidden(False)
    adminLabel.set_hidden(False)

    downArrowImage1.set_hidden(False)
    downArrowImage2.set_hidden(False)
    downArrowImage3.set_hidden(False)


def hideLoginPage():
    """Hides all UI elements from the login page"""
    pickRoleLabel1.set_hidden(True)
    pickRoleLabel2.set_hidden(True)

    defaultLabel.set_hidden(True)
    maintLabel.set_hidden(True)
    adminLabel.set_hidden(True)

    downArrowImage1.set_hidden(True)
    downArrowImage2.set_hidden(True)
    downArrowImage3.set_hidden(True)
    pass
```

```
def buttonA_wasPressed():
    """The 'default' role is selected. If a role has not been previously selected the Sound Puzzle will be setup"""
    global roleSelected
    if not roleSelected:
        roleSelected = True
    pass
btnA.wasPressed(buttonA_wasPressed)


# Maintenance role selected
def buttonB_wasPressed():
    """The 'maintenance' role is selected. If a role has not been previously selected the Flow Puzzle will be setup"""
    global roleSelected
    if not roleSelected:
        roleSelected = True
    pass
btnB.wasPressed(buttonB_wasPressed)


# Admin role selected
def buttonC_wasPressed():
    """The 'admin' role is selected. If a role has not been previously selected the Morse Puzzle will be setup"""
    global roleSelected
    if not roleSelected:
        roleSelected = True
    pass
btnC.wasPressed(buttonC_wasPressed)


setUpLoginPage()
```

```
datetime.py > ...
import unittest
import datetime
import pytz
import utils

# Tests if the date variables are equal to the date imported from the datetime
# package and tests if it is str type after conversion.

class DateTimeEquality(unittest.TestCase):
    def setUp(self):
        self.date = utils.date
        self.str_date = utils.str_date

    def test_assert_equal(self):
        self.assertEqual(self.date, datetime.datetime.now(pytz.timezone('US/Eastern')))

    def test_type(self):
        self.assertIsInstance(self.str_date, str)


if __name__ == '__main__':
    unittest.main()
```

login.py 1 ×

_login.py > ...

```python
import unittest
from Software import Software, Users

# Tests _login_data if dictionary length matches

class Login(unittest.TestCase):
    def setUp(self):
        self.len_login_data = len(Software._LOGIN_DATA)
        self.len_astronaut_login = len(Software._LOGIN_DATA[Users.Astronaut])
        self.len_maintenance_login = len(Software._LOGIN_DATA[Users.Astronaut])
        self.len_admin_login = len(Software._LOGIN_DATA[Users.Astronaut])

    # Tests that there are 3 users in login data dict
    def test_logindata_len(self):
        self.assertTrue(self.len_login_data, 3)

    # Tests that there are 2 fields for each user dict (username and pass)
    def test_users_len(self):
        self.assertTrue(self.len_astronaut_login, 2)
        self.assertTrue(self.len_maintenance_login, 2)
        self.assertTrue(self.len_admin_login, 2)


if __name__ == '__main__':
    unittest.main()
```

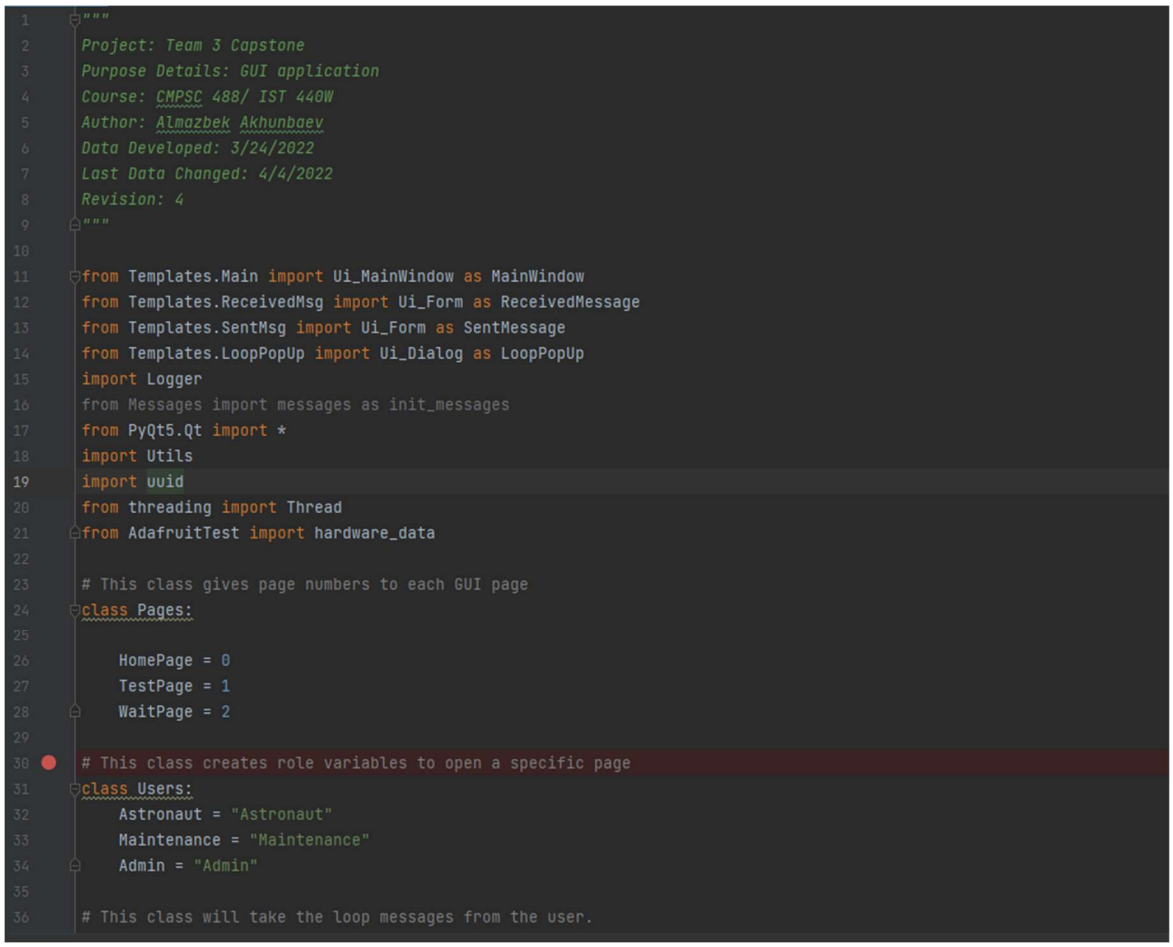
```python
import unittest
from Software import Users

# Tests if user selection matches and if it is of type String

class AstronautType(unittest.TestCase):
    def setUp(self):
        self.choice = Users.Astronaut

    def test_assert_true(self):
        self.assertTrue(type(self.choice) is str)

    def test_user_selection(self):
        self.assertIsInstance(self.choice, str)

    def test_assert_is_with_type(self):
        self.assertIs(type(self.choice), str)

    def test_assert_is(self):
        self.assertIs(self.choice, Users.Astronaut)


class MaintenanceType(unittest.TestCase):
    def setUp(self):
        self.choice = Users.Maintenance

    def test_assert_true(self):
        self.assertTrue(type(self.choice) is str)

    def test_user_selection(self):
        self.assertIsInstance(self.choice, str)

    def test_assert_is_with_type(self):
        self.assertIs(type(self.choice), str)

    def test_assert_is(self):
        self.assertIs(self.choice, Users.Maintenance)
```

```
class AdminType(unittest.TestCase):
    def setUp(self):
        self.choice = Users.Admin

    def test_assert_true(self):
        self.assertTrue(type(self.choice) is str)

    def test_user_selection(self):
        self.assertIsInstance(self.choice, str)

    def test_assert_is_with_type(self):
        self.assertIs(type(self.choice), str)

    def test_assert_is(self):
        self.assertIs(self.choice, Users.Admin)


if __name__ == '__main__':
    unittest.main()
```

pages.py > ...

```python
import unittest
from Software import Pages

# Tests if page selection matches and if it is of type int

class HomePageType(unittest.TestCase):
    def setUp(self):
        self.choice = Pages.HomePage

    def test_assert_true(self):
        self.assertTrue(type(self.choice) is int)

    def test_user_selection(self):
        self.assertIsInstance(self.choice, int)

    def test_assert_is_with_type(self):
        self.assertIs(type(self.choice), int)

    def test_assert_is(self):
        self.assertIs(self.choice, Pages.HomePage)


class TestPageType(unittest.TestCase):
    def setUp(self):
        self.choice = Pages.TestPage

    def test_assert_true(self):
        self.assertTrue(type(self.choice) is int)

    def test_user_selection(self):
        self.assertIsInstance(self.choice, int)

    def test_assert_is_with_type(self):
        self.assertIs(type(self.choice), int)

    def test_assert_is(self):
        self.assertIs(self.choice, Pages.TestPage)

if __name__ == '__main__':
    unittest.main()
```

23.5 main.py



```
"""
Project: Team 3 Capstone
Purpose Details: GUI application
Course: CMPSC 488/ IST 440W
Author: Almazbek Akhunbaev
Data Developed: 3/24/2022
Last Data Changed: 4/4/2022
Revision: 4
"""

from Templates.Main import Ui_MainWindow as MainWindow
from Templates.ReceivedMsg import Ui_Form as ReceivedMessage
from Templates.SentMsg import Ui_Form as SentMessage
from Templates.LoopPopUp import Ui_Dialog as LoopPopUp
import Logger
from Messages import messages as init_messages
from PyQt5.Qt import *
import Utils
import uuid
from threading import Thread
from AdafruitTest import hardware_data

# This class gives page numbers to each GUI page
class Pages:

    HomePage = 0
    TestPage = 1
    WaitPage = 2

# This class creates role variables to open a specific page
class Users:
    Astronaut = "Astronaut"
    Maintenance = "Maintenance"
    Admin = "Admin"

# This class will take the loop messages from the user.
```

```
35
36  # This class will take the loop messages from the user.
37  class LoopMessage(QDialog, LoopPopUp):
38
39      def __init__(self, parent=None):
40          # This is a self function that will run automatically
41          super(LoopMessage, self).__init__(parent)
42          self.setupUi(self)
43          self.setWindowTitle("Loop")
44          self.setWindowFlag(Qt.FramelessWindowHint)
45          self.setAttribute(Qt.WA_TranslucentBackground)
46          self.c_number_messages = None
47          self.send.clicked.connect(self._send)
48          self.close_btn.clicked.connect(self.close)
49          self.init_graphics()
50
51      def set_blur(self, widget):
52          # This class add a blureeffect to the GUI
53          blureffect = QGraphicsDropShadowEffect()
54          blureffect.setBlurRadius(20)
55          blureffect.setOffset(2, 2)
56          blureffect.setColor(QColor(120, 111, 100))
57          widget.setGraphicsEffect(blureffect)
58
59      def init_graphics(self):
60          self.set_blur(self.container)
61
62      def center(self):
63          qr = self.frameGeometry()
64          cp = QDesktopWidget().availableGeometry().center()
65          qr.moveCenter(cp)
66          self.move(qr.topLeft())
67
68      def mousePressEvent(self, event):
69          self.oldPos = event.globalPos()
```

```
70
71      def mouseMoveEvent(self, event):
72          if not self.isMaximized():
73              delta = QPoint(event.globalPos() - self.oldPos)
74              self.move(self.x() + delta.x(), self.y() + delta.y())
75              self.oldPos = event.globalPos()
76
77      def _send(self):
78          nb = self.number_msgs.value()
79          if nb >= 1:
80              self.c_number_messages = nb
81              self.close()
82
83      def res(self):
84          return self.c_number_messages
85
86
87  class SendMessageWidget(QWidget, SentMessage):
88      # This class is responsible for sending messages
89      def __init__(self, parent=None):
90          super(SendMessageWidget, self).__init__(parent)
91          self.setupUi(self)
92          self.close_btn.setVisible(False)
93          self.close_btn.clicked.connect(lambda: self.callback(self))
94          self.id = None
95          self._init_widget()
96
97      def _init_widget(self):
98          self.sent_message.setWordWrap(True)
99
100     def set_date(self, message_date):
101         self.date.setText(message_date)
102
103     def set_message(self, msg):
104         self.sent_message.setText(msg)
105
```

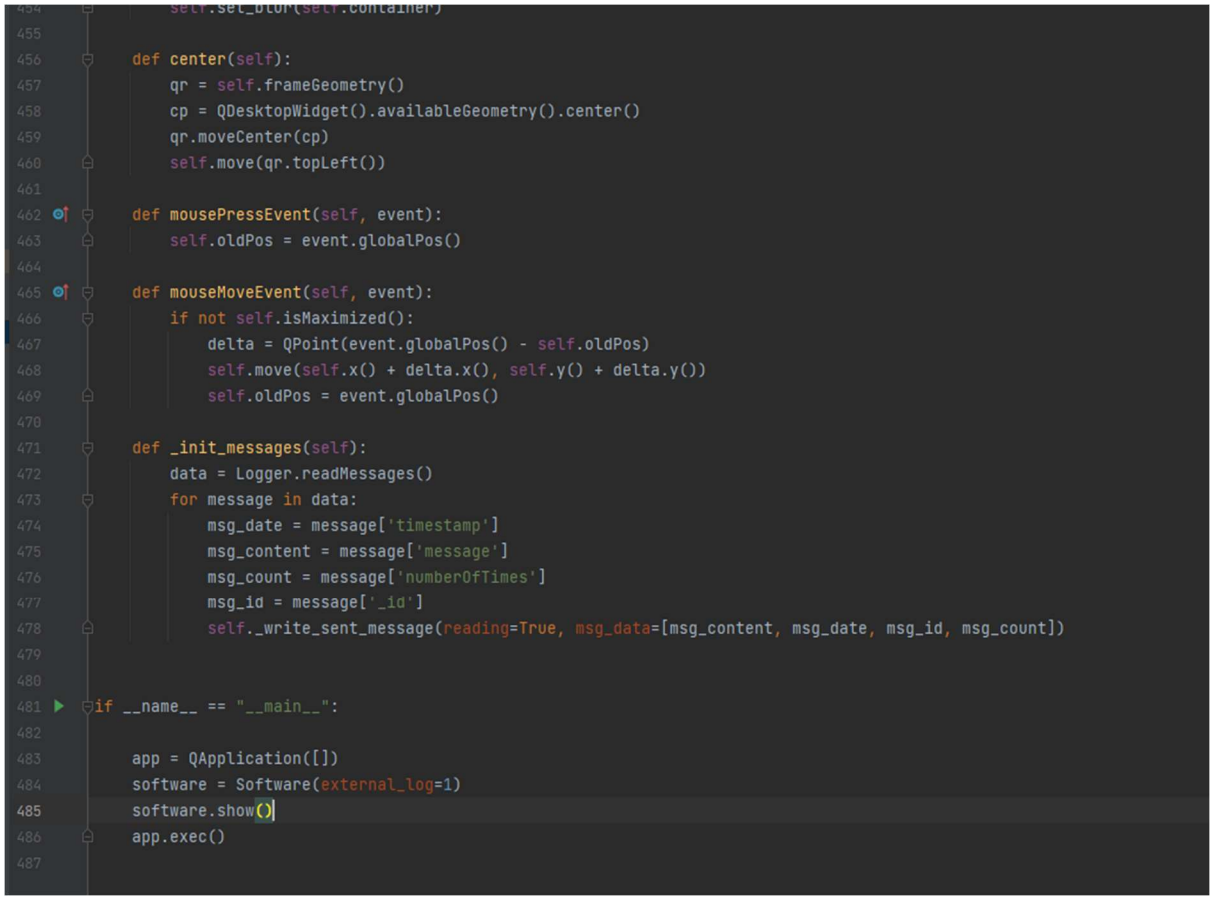
```
102
103     def set_message(self, msg):
104         self.sent_message.setText(msg)
105
106     def set_id(self, id):
107         self.id = id
108
109     def activate_close(self):
110         self.close_btn.setVisible(True)
111
112     def callback(self, *args, **kwargs):
113         print("This is the Initial callback.")
114
115
116 class ReceiveMessageWidget(QWidget, ReceivedMessage):
117     # This class is responsible for receiving messages
118     def __init__(self, parent=None):
119         super(ReceiveMessageWidget, self).__init__(parent)
120         self.setupUi(self)
121         self.close_btn.clicked.connect(lambda: self.callback(self))
122         self.id = None
123         self._init_widget()
124
125     def _init_widget(self):
126         self.receivedmessage.setWordWrap(True)
127         self.close_btn.setVisible(False)
128
129     def activate_close(self):
130         self.close_btn.setVisible(True)
131
132     def set_date(self, message_date):
133         self.date.setText(message_date)
```

```
133         self.date.setText(message_date)
134
135     def set_message(self, msg):
136         self.receivedmessage.setText(msg)
137
138     def callback(self, *args, **kwargs):
139         print("This is the Initial callback.")
140
141
142 class Software(QMainWindow, MainWindow):
143     hardware_signal = pyqtSignal(int)
144     _LOGIN_DATA = {
145         Users.Astronaut: {
146             "username": "astr", "pwd": "astr"
147         },
148         Users.Maintenance: {
149             "username": "man", "pwd": "man"
150         },
151         Users.Admin: {
152             "username": "admin", "pwd": "admin"
153         }
154     }
155
156     _DEBUG = True
157
158     def __init__(self, external_log=None, parent=None):
159         super(Software, self).__init__(parent)
160         self.setupUi(self)
161         self.setWindowTitle("Astro Message")
162         self.setWindowFlag(Qt.FramelessWindowHint)
163         self.setAttribute(Qt.WA_TranslucentBackground)
164         self.beacon_state = False
165         self.logged_user = None
```

```
159         super(Software, self).__init__(parent)
160         self.setupUi(self)
161         self.setWindowTitle("Astro Message")
162         self.setWindowFlag(Qt.FramelessWindowHint)
163         self.setAttribute(Qt.WA_TranslucentBackground)
164         self.beacon_state = False
165         self.logged_user = None
166         self._sent_messages = {}
167         self._sent_messages_array = []
168         self._received_messages = []
169         self.msg_pointer = []
170         self.init_graphics()
171         self.oldPos = self.pos()
172         self._init_app()
173         self.listen_to_hardware()
174
175         # self._external_login(external_log)
176
177     def _external_login(self, external_log):
178         if external_log is not None:
179             if external_log == 3:
180                 self._access_to_system(Users.Astronaut)
181             elif external_log == 1:
182                 self._access_to_system(Users.Admin)
183             elif external_log == 2:
184                 self._access_to_system(Users.Maintenance)
185
186             if self._DEBUG:
187                 self._init_messages()
188
189     def listen_to_hardware(self):
190         thread = Thread(target=hardware_data, args=[self.hardware_signal])
191         thread.daemon = True
192         thread.start()
```

```
188
189     def listen_to_hardware(self):
190         thread = Thread(target=hardware_data, args=[self.hardware_signal])
191         thread.daemon = True
192         thread.start()
193
194     def _create_msg(self, msg, date, nb, msg_id=None):
195         if msg_id is None:
196             msg_id = str(uuid.uuid1())
197
198         message = {
199             "id": msg_id,
200             "time": date,
201             "numberOfTimes": nb,
202             "message": msg
203         }
204
205         return message
206
207     def _send_loop(self):
208         import traceback
209         try:
210             wind = LoopMessage()
211             wind.exec()
212             res = wind.res()
213             if res is not None:
214                 self._write_sent_message(res)
215         except:
216             print(traceback.format_exc())
217
218     def _toggle_beacon(self):
219         self.beacon_state = not self.beacon_state
220         print(self.beacon_state)
221
222     def _login(self):
```

```
                self._write_sent_message(res)
            except:
                print(traceback.format_exc())

    def _toggle_beacon(self):
        self.beacon_state = not self.beacon_state
        print(self.beacon_state)

    def _login(self):
        user_name = self.user_name_field.text()
        pwd = self.password_field.text()
        for user_type in self._LOGIN_DATA.keys():
            user = self._LOGIN_DATA[user_type]
            luser_name = user['username']
            lpwd = user['pwd']
            if user_name == luser_name and pwd == lpwd:
                self._access_to_system(user_type)
                break

    def _access_to_system(self, user):
        self._select_page(Pages.HomePage)
        self._manage_menu_buttons(True, user)
        self.logged_user = user

    def _write_received_message(self, msg):
        receive_form = ReceiveMessageWidget(self.messages)
        receive_form.set_date(Utils.actual_date())
        receive_form.set_message(msg)
        receive_form.callback = self._delete_message
        self.verticalLayout_6.addWidget(receive_form)

    def _store_message(self, msg, internal=False):
        self._sent_messages[msg['id']] = (msg)
        self._sent_messages_array = list(self._sent_messages.values())
        print(self._sent_messages_array)
        if not internal:
```



```
        self.set_blur(self.container)

    def center(self):
        qr = self.frameGeometry()
        cp = QDesktopWidget().availableGeometry().center()
        qr.moveCenter(cp)
        self.move(qr.topLeft())

    def mousePressEvent(self, event):
        self.oldPos = event.globalPos()

    def mouseMoveEvent(self, event):
        if not self.isMaximized():
            delta = QPoint(event.globalPos() - self.oldPos)
            self.move(self.x() + delta.x(), self.y() + delta.y())
            self.oldPos = event.globalPos()

    def _init_messages(self):
        data = Logger.readMessages()
        for message in data:
            msg_date = message['timestamp']
            msg_content = message['message']
            msg_count = message['numberOfTimes']
            msg_id = message['_id']
            self._write_sent_message(reading=True, msg_data=[msg_content, msg_date, msg_id, msg_count])


if __name__ == "__main__":

    app = QApplication([])
    software = Software(external_log=1)
    software.show()
    app.exec()

```

23.5 Logger.py

```
"""
Project: Team 3 Capstone
Purpose Details: GUI application
Course: CMPSC 488/ IST 440W
Authors: Almazbek Akhunbaev, Matthew Coutts, and Bradley Whitebread
Data Developed: 3/24/2022
Last Data Changed: 4/4/2022
Revision: 4
"""

import requests, datetime
import json
import pprint


def log_event(event,
              status='testing',
              duration='test',
              collection='logs'):
    # host = 'localhost'
    # port = 5000
    # host = '0.0.0.0'
    # host = '192.168.1.196'
    host = '10.0.0.188'
    port = 49153

    log_event = {
        "app": event,
        "status": status,
        "time": str(datetime.datetime.now()),
        "duration": duration
    }

    req = requests.post('http://' + host + ':' + str(port) + '/' + collection, data=log_event)
    print(type(req.json()))
    print(req.json())
```

deleteMessage()

```
26
27      log_event = {
28          "app": event,
29          "status": status,
30          "time": str(datetime.datetime.now()),
31          "duration": duration
32      }
33
34      req = requests.post('http://' + host + ':' + str(port) + '/' + collection, data=log_event)
35      print(type(req.json()))
36      print(req.json())
37
38      res_status = req.json()['_status']
39
40      # eTag = req.json()['_etag']
41      # res_id = req.json()['_id']
42      if (res_status == 'OK'):
43          return True
44
45      else:
46          print(res_status)
47          return False
48
49
50  def saveMessage(message, numberOfMessages=1, collection='messages'):
51      # This function will save sent messages to the database
52      # host = '0.0.0.0'
```

```python

def saveMessage(message, numberOfMessages=1, collection='messages'):
    # This function will save sent messages to the database
    # host = '0.0.0.0'
    # host = '192.168.1.196'
    host = '10.0.0.188'
    port = 49153

    ms = {
        "message": message,
        "numberOfTimes": numberOfMessages,
        "timestamp": str(datetime.datetime.now()),
        "sent": False
    }

    req = requests.post('http://' + host + ':' + str(port) + '/' + collection, data=ms)
    print(type(req.json()))
    print(req.json())

    res_status = req.json()['_status']
    # eTag = req.json()['_etag']
    # res_id = req.json()['_id']
    if (res_status == 'OK'):
        return True

    else:
        print(res_status)
        return False

```

```
78
79  def readMessages():
80      # This function will read sent messages from the database to the GUI app
81      # host = '0.0.0.0'
82      # host = '192.168.1.196'
83      host = '10.0.0.188'
84      port = 49153
85      collection = "messages"
86      print('http://' + host + ':' + str(port) + '/' + collection)
87
88      req = requests.get('http://' + host + ':' + str(port) + '/' + collection)
89      reqData = json.loads(req.text)
90      # print(json.dumps(reqData, indent=4, sort_keys = "timestamp"))
91      data = json.dumps(reqData, indent=4, sort_keys="timestamp")
92      messages = json.loads(data)
93      return messages['_items']
94
95
96  def deleteMessage(message_id):
97      # This function will delete sent messages from the database
98      # host = "0.0.0.0"
99      # host = '192.168.1.196'
100     host = '10.0.0.188'
101     port = 49153
102     collection = "messages"
103
104     url = 'http://' + host + ':' + str(port) + '/' + collection + '/'
105
106     req1 = requests.get(url + message_id)
107
108     if req1.status_code == 200:
109         req2 = requests.delete(url, data={"_id": message_id})
110
111
112 if __name__ == '__main__':
        deleteMessage()
```

23.5 AdafruitTest.py

```
"""
Purpose Details: GUI application
Course: CMPSC 488/ IST 440W
Authors: Almazbek Akhunbaev, Matthew Coutts, and Bradley Whitebread
Data Developed: 3/24/2022
Last Data Changed: 4/4/2022
Revision: 4
"""

from Adafruit_IO import Client
from datetime import datetime, timezone
import time


def hardware_data(user_signal):
    # This function will be use to open a specific page for a specific role
    aio = Client("bew5294", "aio_pLPP82smfxwykOdHBbsug5RXEBM5")

    currentTime = str(datetime.now(timezone.utc))
    currentTime = currentTime[:19]
    currentTime = datetime.strptime(currentTime, "%Y-%m-%d %H:%M:%S")

    while True:
        data = aio.data('capstone-morse-communicator')
        lastMessageTime = str(data[0].created_at)
        lastMessageTime = lastMessageTime[:10] + " " + lastMessageTime[11:19]

        lastMessageTime = datetime.strptime(lastMessageTime, "%Y-%m-%d %H:%M:%S")

        if currentTime == max(currentTime, lastMessageTime):
            # print("old message")
            pass
        else:
            # print("new message:", data[0].value)
            if data[0].value == "admin":
                # redirect to admin page
                user_signal.emit(1)
            elif data[0].value == "maintenance":
                # redirect to maintenance page
                user_signal.emit(2)
```

23.6 Morse Code Puzzle

```
"""
Project: Team 3 Capstone
Purpose Details: Implementation of the Sound Puzzle
Course: CMPSC 488/ IST 440W
Author: Matthew Coutts & Bradley Whitebread
Date Developed: 3/23/2022
Last Date Changed: 3/31/2022
Revision: 4
"""

from m5stack import *
from m5stack_ui import *
from uiflow import *
```

```
import os
import time

screen = M5Screen()
screen.clean_screen()
screen.set_screen_bg_color(0xFFFFFF)

soundList = None
answer = None
sound1 = None
sound2 = None
sound3 = None

touch_button0 = M5Btn(text='1', x=125, y=40, w=70, h=30, bg_c=0xFFFFFF, text_c=0x000000, font=FONT_MONT_14,
parent=None)
label0 = M5Label('_', x=5, y=210, color=0x000, font=FONT_MONT_14, parent=None)
touch_button1 = M5Btn(text='2', x=125, y=105, w=70, h=30, bg_c=0xFFFFFF, text_c=0x000000,
font=FONT_MONT_14,
                      parent=None)
touch_button2 = M5Btn(text='3', x=125, y=165, w=70, h=30, bg_c=0xFFFFFF, text_c=0x000000,
font=FONT_MONT_14,
                      parent=None)
label1 = M5Label('?', x=149, y=102, color=0x000, font=FONT_MONT_36, parent=None)

import random


def touch_button1_pressed():
    global soundList, answer, sound1, sound2, sound3
    if 2 == answer:
        screen.set_screen_bg_color(0x33ff33)
    else:
        screen.set_screen_bg_color(0xff0000)
    pass


touch_button1.pressed(touch_button1_pressed)


def touch_button2_pressed():
    global soundList, answer, sound1, sound2, sound3
    if 3 == answer:
        screen.set_screen_bg_color(0x33ff33)
    else:
        screen.set_screen_bg_color(0xff0000)
    pass


touch_button2.pressed(touch_button2_pressed)


# This section changes the screen color if the user got the right answer
#
# RED = wrong
```

```
# Green = right
def touch_button0_pressed():
    global soundList, answer, sound1, sound2, sound3
    if 1 == answer:
        screen.set_screen_bg_color(0x33ff33)
    else:
        screen.set_screen_bg_color(0xff0000)
    pass


touch_button0.pressed(touch_button0_pressed)

soundList = os.listdir('/sd//sounds')
# these are our touch buttons when the user solves the puzzle
#
touch_button0.set_pos(340, 40)
touch_button1.set_pos(340, 105)
touch_button2.set_pos(340, 165)
# Label 0 = 1
# Label 1 = 2
# Label 2 = 3
label0.set_pos(340, 105)
label0.set_text('')
# here we randomly assign the vibrations to random speeds
sound1 = random.randint(1, 3)
sound2 = random.randint(1, 3)
while sound1 == sound2:
    # here we randomly assign the vibrations to random speeds
    sound2 = random.randint(1, 3)
sound3 = random.randint(1, 3)
while sound2 == sound3 or sound1 == sound3:
    sound3 = random.randint(1, 3)
label1.set_pos(125, 105)
label1.set_text('1')
speaker.playWAV(((str('/sd/sounds/') + str(soundList[0]))))
wait(0.5)
label1.set_pos(340, 105)
wait(0.5)
label1.set_pos(125, 105)
label1.set_text('2')
speaker.playWAV('/sd/sounds/beep-07a (1).wav')
wait(1)
label1.set_pos(340, 105)
wait(0.5)
label1.set_pos(125, 105)
label1.set_text('3')
speaker.playWAV(((str('/sd/sounds/') + str(soundList[2]))))
wait(0.5)
label1.set_pos(340, 105)
wait(0.5)
label1.set_pos(125, 105)
label1.set_text('?')
if random.randint(1, 3) == 1:
    speaker.playWAV(((str('/sd/sounds/') + str(soundList[0]))))
```

```
    answer = 1
elif random.randint(1, 3) == 2:
    speaker.playWAV('/sd/sounds/beep-07a (1).wav')
    answer = 2
elif random.randint(1, 3) == 3:
    speaker.playWAV(((str('/sd/sounds/') + str(soundList[2]))))
    answer = 3
else:
    label1.set_text('ERROR')
label1.set_pos(340, 105)
touch_button0.set_pos(125, 40)
touch_button1.set_pos(125, 105)
touch_button2.set_pos(125, 165)
```

24.0 Testing Procedures

24.1 UAT

Item No.	Functional Requirement	Priority	Result	Comments
1	The system will translate inputted English messages into Morse code.	HIGH	Completed	
2	The system will translate received Morse code messages into English.	LOW	Partial	We cut the part of the system that would receive due to time limitations.
3	The application will contain a message log that will be accessible by users.	HIGH	Completed	
4	The system will output a random word or phrase in morse code that must be matched to log in.	HIGH	Completed	
5	The system will vary the number of options depending on the user level. (3 for the basic user, 5 for maintenance user, 7 for admin user)	LOW	Partial	Instead of different difficulties we implemented three different types of puzzles. Those being the Sound, Flow, and Morse puzzles.
6	The system shall allow the user two tries before the Morse code phrase updates.	LOW	Partial	We opted to only have one chance per word in the Morse puzzle. This is because it is the hardest puzzle for the highest access and there are only three options. If given two chances someone could guess randomly and proceed most of the time
7	The application will allow a user with admin privileges to delete messages.	HIGH	Completed	
8	The application will display what morse code would be broadcast for a given English text input.	HIGH	Completed	
9	The application will display what English text would be translated for a given Morse code input.	HIGH	Completed	
10	The system will have an emergency button which once pressed will allow users to send one message.	LOW	Unaddressed	We did not have enough time to implement this feature. And since it wasn't as important as sending messages and the puzzles it was cut
11	The system will have an emergency button which once pressed will allow users to only view the past five messages.	LOW	Unaddressed	Also cut due to time constraints.

24.2 Security Testing

Bandit is a tool for testing security in Python applications. It is designed to be simple to use, yet powerful enough to test the most complex interactions between code. Bandit provides both unit tests and integration tests and works proficiently with unit tests written in Python and third-party libraries. We used Bandit to test our .py files to determine the security issues present within our code (Bandit Developers Revision).

Driver File Main

filename	test_name	test_id	issue_severity	issue_confidence	issue_cwe	issue_text	line_number	col_offset	line_range	more_ir
main.py	blacklist	B311	LOW	HIGH	https://cwe.	Standard pseu	180	33	[180]	https://
main.py	blacklist	B311	LOW	HIGH	https://cwe.	Standard pseu	192	17	[192]	https://
main.py	blacklist	B311	LOW	HIGH	https://cwe.	Standard pseu	193	17	[193]	https://
main.py	blacklist	B311	LOW	HIGH	https://cwe.	Standard pseu	195	21	[195]	https://
main.py	blacklist	B311	LOW	HIGH	https://cwe.	Standard pseu	197	17	[197]	https://
main.py	blacklist	B311	LOW	HIGH	https://cwe.	Standard pseu	199	21	[199]	https://
main.py	try_except_pass	B110	LOW	HIGH	https://cwe.	Try, Except, Pa	271	8	[271]	https://
main.py	try_except_pass	B110	LOW	HIGH	https://cwe.	Try, Except, Pa	301	8	[301]	https://
main.py	try_except_pass	B110	LOW	HIGH	https://cwe.	Try, Except, Pa	331	8	[331]	https://
main.py	try_except_pass	B110	LOW	HIGH	https://cwe.	Try, Except, Pa	503	8	[503]	https://
main.py	blacklist	B311	LOW	HIGH	https://cwe.	Standard pseu	831	35	[831]	https://
main.py	blacklist	B311	LOW	HIGH	https://cwe.	Standard pseu	843	17	[843]	https://
main.py	blacklist	B311	LOW	HIGH	https://cwe.	Standard pseu	844	17	[844]	https://
main.py	blacklist	B311	LOW	HIGH	https://cwe.	Standard pseu	846	21	[846]	https://
main.py	blacklist	B311	LOW	HIGH	https://cwe.	Standard pseu	848	17	[848]	https://
main.py	blacklist	B311	LOW	HIGH	https://cwe.	Standard pseu	850	21	[850]	https://
main.py	try_except_pass	B110	LOW	HIGH	https://cwe.	Try, Except, Pa	891	8	[891]	https://
main.py	try_except_pass	B110	LOW	HIGH	https://cwe.	Try, Except, Pa	921	8	[921]	https://
main.py	try_except_pass	B110	LOW	HIGH	https://cwe.	Try, Except, Pa	951	8	[951]	https://

Flow Puzzle

filename	test_name	test_id	issue_sev	issue_con	issue_cwe	issue_text	line_number	col_offset	line_range	mo
FlowPuzzle.py	try_except_pass	B110	LOW	HIGH	https://cwe.n	Try, Except, Pass detected.	192	8	[192]	http

GUI Main

filename	test_name	test_id	issue_severity	issue_confidence	issue_cwe	issue_text	line_number	col_offset	line_range	more
main.py	try_except_pass	B110	LOW	HIGH	https://cwe.mi	Try, Except, Pass detect	433	12	[433]	https

Morse Puzzle

filename	test_name	test_id	issue_sev	issue_con	issue_cwe	issue_text	line_number	col_offset	line_range	more_
MorsePuzzle.py	blacklist	B311	LOW	HIGH	https://cwe.r	Standard pse	63	35	[63]	https:/
MorsePuzzle.py	blacklist	B311	LOW	HIGH	https://cwe.r	Standard pse	75	17	[75]	https:/
MorsePuzzle.py	blacklist	B311	LOW	HIGH	https://cwe.r	Standard pse	76	17	[76]	https:/
MorsePuzzle.py	blacklist	B311	LOW	HIGH	https://cwe.r	Standard pse	78	21	[78]	https:/
MorsePuzzle.py	blacklist	B311	LOW	HIGH	https://cwe.r	Standard pse	80	17	[80]	https:/
MorsePuzzle.py	blacklist	B311	LOW	HIGH	https://cwe.r	Standard pse	82	21	[82]	https:/
MorsePuzzle.py	try_except_pass	B110	LOW	HIGH	https://cwe.r	Try, Except, P	123	8	[123]	https:/
MorsePuzzle.py	try_except_pass	B110	LOW	HIGH	https://cwe.r	Try, Except, P	153	8	[153]	https:/
MorsePuzzle.py	try_except_pass	B110	LOW	HIGH	https://cwe.r	Try, Except, P	183	8	[183]	https:/

Sound Puzzle

filename	test_name	test_id	issue_severity	issue_con	issue_cwe	issue_text	line_number	col_offset	line_range	more_in
SoundPuzzle.py	blacklist	B311	LOW	HIGH	https://cw	Standard pseudo-random generators a	76	13	[76]	https://l
SoundPuzzle.py	blacklist	B311	LOW	HIGH	https://cw	Standard pseudo-random generators a	77	13	[77]	https://l
SoundPuzzle.py	blacklist	B311	LOW	HIGH	https://cw	Standard pseudo-random generators a	78	13	[78]	https://l
SoundPuzzle.py	blacklist	B311	LOW	HIGH	https://cw	Standard pseudo-random generators a	105	3	[105]	https://l
SoundPuzzle.py	blacklist	B311	LOW	HIGH	https://cw	Standard pseudo-random generators a	111	5	[111]	https://l
SoundPuzzle.py	blacklist	B311	LOW	HIGH	https://cw	Standard pseudo-random generators a	117	5	[117]	https://l

Testing Procedures Conflicts

Unfortunately, due to the capabilities of the M5 Stack software and hardware we were unable to complete the following tests: Integration Testing, System Testing, Smoke Testing, and Load Testing. In addition, some of the security tests performed by Bandit may be false positives leading to inaccurate results.

25.0 Test Plan

25.1 Login Page

Test No:	1	Steps	Description	Expected Result
Priority	High	Step 1	Turn on M5 Stack	M5 Stack turns on and displays the login page
Execution Result		Step 2	Select the Default role (leftmost button on M5)	The sound puzzle will start
Purpose	Default Role Selected			
Material Needed	M5 Stack			
Created By	Bradley Whitebread			

Test No:	2	Steps	Description	Expected Result
Priority	High	Step 1	Turn on M5 Stack	M5 Stack turns on and displays the login page
Execution Result		Step 2	Select the Maintenance role (middle button on M5)	The flow puzzle will start
Purpose	Maintenance Role Selected			
Material Needed	M5 Stack			
Created By	Bradley Whitebread			

Test No:	3	Steps	Description	Expected Result
Priority	High	Step 1	Turn on M5 Stack	M5 Stack turns on and displays the login page
Execution Result		Step 2	Select the Admin role (rightmost button on M5)	The morse puzzle will start
Purpose	Admin Role Selected			
Material Needed	M5 Stack			
Created By	Bradley Whitebread			

25.2 Sound Puzzle

Test No:	4	Steps	Description	Expected Result
Priority	High	Step 1	Turn on M5 with program installed	M5 boots and program initializes
Execution Result		Step 2	User listens to sounds	Sounds will cycle through with 1, 2, 3, ?
Purpose	If M5 plays sounds	Step 3	All sounds play through	Sounds will commence and screen will change
Material Needed	M5, SD card inserted			
Created By	Matt Coutts			

Test No:	5	Steps	Description	Expected Result
Priority	High	Step 1	Turn on M5 with program installed	M5 boots and program initializes
Execution Result		Step 2	User listens to sounds	Sounds will cycle through with 1, 2, 3, ?
Purpose	Touch screen is working/guess sound	Step 3	All sounds play through	Sounds will finish and screen will change
Material Needed	M5, SD card inserted	Step 4	Touch screen is activated	User can press 3 buttons to guess mystery sound
Created By	Matt Coutts	Step 5	User presses button	M5 should receive input
		Step 6	Input received	Displays green (correct guess) or red

Test No:	6	Steps	Description	Expected Result
Priority	Medium	Step 1	Turn on M5 with program installed	M5 boots and program initializes
Execution Result		Step 2	User listens to sounds	Sounds will cycle through with 1, 2, 3, ?
Purpose	See if guess function works with random input	Step 3	All sounds play through	Sounds will commence and screen will change
Material Needed	M5, SD card inserted	Step 4	Touch screen is activated	User can press 3 buttons to guess mystery sound
Created By	Matt Coutts	Step 5	Press buttons repeatedly	M5 should receive first input

Test No:	7	Steps	Description	Expected Result
Priority	High	Step 1	Turn on M5 with program installed	M5 boots and program initializes
Execution Result		Step 2	User listens to sounds	Sounds will cycle through with 1, 2, 3, ?
Purpose	Tapping ouside of buttons cause malfunction	Step 3	All sounds play through	Sounds will commence and screen will change
Material Needed	M5, SD card inserted	Step 4	Touch screen is activated	User can press 3 buttons to guess mystery sound
Created By	Matt Coutts	Step 5	Press touchscreen outside of buttons	M5 should not do anything and wait for input

Test No:	8	Steps	Description	Expected Result
Priority	High	Step 1	Turn on M5 with program installed	M5 boots and program initializes
Execution Result		Step 2	User listens to sounds	Sounds will cycle through with 1, 2, 3, ?
Purpose	Tapping screen when sound is playing	Step 3	Touch screen while sounds are playing	Sounds should advance automatically
Material Needed	M5, SD card inserted	Step 4	Sounds play through	Guess screen appears
Created By	Matt Coutts	Step 5	User guesses	Result is marked wrong or right

25.3 Flow Puzzle

Test No:	9	Steps	Description	Expected Result
Priority	High	Step 1	Turn on M5 Stack	M5 Stack turns on and displays the login page
Execution Result		Step 2	Select maintenance role (middle button)	Loads flow puzzle
Purpose	Flow puzzle loads UI elements			
Material Needed	M5 Stack			
Created By	Bradley Whitebread			

Test No:	10	Steps	Description	Expected Result
Priority	Medium	Step 1	Turn on M5 Stack	M5 Stack turns on and displays the login page
Execution Result		Step 2	Select maintenance role (middle button)	Loads flow puzzle
Purpose	Check Puzzle function works correctly	Step 3	Complete flow puzzle	Should advance to the broadcasting screen
Material Needed	M5 Stack			
Created By	Bradley Whitebread			

Test No:	11	Steps	Description	Expected Result
Priority	High	Step 1	Turn on M5 Stack	M5 Stack turns on and displays the login page
Execution Result		Step 2	Select maintenance role (middle button)	Loads flow puzzle
Purpose	Buttons color change to the last pressed root	Step 3	Press and hold a root node	Button should appear white while being pressed
Material Needed	M5 Stack	Step 4	Drag finger to any non root node (a node without a black circle)	the non root node should change color to the last root node
Created By	Bradley Whitebread			

Test No:	12	Steps	Description	Expected Result
Priority	High	Step 1	Turn on M5 Stack	M5 Stack turns on and displays the login page
Execution Result		Step 2	Select maintenance role (middle button)	Loads flow puzzle
Purpose	Buttons do not change color if finger released since touching the root node	Step 3	Press and release a root node	Button should appear white while being pressed
Material Needed	M5 Stack	Step 4	Press any non root node (a node without a black circle)	The non root node's color shouldn't change
Created By	Bradley Whitebread			

25.4 Morse Puzzle

Test No:	14	Steps	Description	Expected Result
Priority	High	Step 1	Turn on M5 Stack	M5 Stack turns on and displays the login page
Execution Result		Step 2	Select the Admin role (rightmost button on M5)	A word in morse code will be outputted and three word option buttons will be displayed
Purpose	Loads Morse Puzzle			
Material Needed	M5 Stack			
Created By	Bradley Whitebread			

Test No:	15	Steps	Description	Expected Result
Priority	High	Step 1	Turn on M5 Stack	M5 Stack turns on and displays the login page
Execution Result		Step 2	Select the Admin role (rightmost button on M5)	A word in morse code will be outputted and three word option buttons will be displayed
Purpose	Outputs Morse Code			
Material Needed	M5 Stack			
Created By	Bradley Whitebread			

Test No:	16	Steps	Description	Expected Result
Priority	High	Step 1	Turn on M5 Stack	M5 Stack turns on and displays the login page
Execution Result		Step 2	Select the Admin role (rightmost button on M5)	A word in morse code will be outputted and three word option buttons will be displayed
Purpose	Button options display			
Material Needed	M5 Stack			
Created By	Bradley Whitebread			

25.5 Broadcast Message

Test No:	17	Steps	Description	Expected Result
Priority	High	Step 1	Turn on M5 Stack	M5 Stack turns on and displays the login page
Execution Result		Step 2	Select any role	The puzzle for the corresponding role should load
Purpose	Broadcast message via the M5 Stack	Step 3	Complete the puzzle to log into the application	Application home page should load on the raspberry pi
Material Needed	M5 Stack and Raspberry Pi	Step 4	Type a message and click the send button on the application	Message should begin braodcasting on the M5Stack sensors
Created By	Bradley Whitebread			

Test No:	18	Steps	Description	Expected Result
Priority	High	Step 1	Turn on M5 Stack	M5 Stack turns on and displays the login page
Execution Result		Step 2	Select any role	The puzzle for the corresponding role should load
Purpose	Broadcast message loop via M5 Stack	Step 3	Complete the puzzle to log into the application	Application home page should load on the raspberry pi
Material Needed	M5 Stack and Raspberry Pi	Step 4	Type a message and click the loop button on the application	A pop-up window should display asking for the number of times the message should be broadcast
Created By	Bradley Whitebread	Step 5	Enter a number and click OK	The message should broadcast and repeat itself for the number of times selected

25.6 GUI Application: Home Page

Test No:	19	Steps	Description	Expected Result
Priority	High	Step 1	Login as an admin user	The Home page of the GUI Application should be open on the Raspberry Pi
Execution Result		Step 2	Click the delete button to delete a message	The message next to the delete button should be removed from the screen and the database
Purpose	Delete the sent messages			
Material Needed	Raspberry Pi, M5 Stack			
Created By	Almazbek Akhunbaev			

Test No:	20	Steps	Description	Expected Result
Priority	High	Step 1	Login as any user	The Home page of the GUI Application should be open on the Raspberry Pi
Execution Result		Step 2	Enter a message in the text box	The text will be in the textbox
Purpose	Send messages	Step 3	Click the send button	The textbox will be cleared, the message will be displayed with other sent messages, and the message will be broadcast via the M5 Stack
Material Needed	Raspberry Pi			
Created By	Almazbek Akhunbaev			

Test No:	21	Steps	Description	Expected Result
Priority	High	Step 1	Login as any user	The Home page of the GUI Application should be open on the Raspberry Pi
Execution Result		Step 2	Enter a message in the text box	The text will be in the textbox
Purpose	Loop a message button	Step 3	Click the loop button	A pop-up will appear where the user is prompted to enter the number of times to repeat a message
Material Needed	Raspberry Pi	Step 4	Enter the number of times to loop a message	The textbox will be cleared, the message will be displayed with other sent messages, and the message will be broadcast via the M5 Stack the number of times entered
Created By	Almazbek Akhunbaev			

25.7 GUI Application: Test Messages Page

Test No:	22	Steps	Description	Expected Result
Priority	High	Step 1	Login as a maintenance user	The Home page of the GUI Application should be open on the Raspberry Pi
Execution Result		Step 2	Click the "Test Message" tab	The Test Message page will be displayed
Purpose	Translate English Button	Step 3	Enter "SOS" in the left-hand textbox	SOS will be in the left textbox
Material Needed	Raspberry Pi, M5 Stack	Step 4	Click the "Translate English"	The following will be displayed in the right textbox "... --- ..."
Created By	Almazbek Akhunbaev			

Test No:	23	Steps	Description	Expected Result
Priority	High	Step 1	Login as a maintenance user	The Home page of the GUI Application should be open on the Raspberry Pi
Execution Result		Step 2	Click the "Test Message" tab	The Test Message page will be displayed
Purpose	Translate Morse Code Button	Step 3	Enter "... --- ..." in the right-hand textbox	... --- ... will be in the right textbox
Material Needed	Raspberry Pi, M5 Stack	Step 4	Click the "Translate Morse Code"	The following will be displayed in the left textbox "SOS"
Created By	Almazbek Akhunbaev			

26.0 Testing Incidents and Summary Reports

Test No.	Pass/ Fail	Defects/ Deviations/ Problems
1	Pass	None
2	Pass	None
3	Pass	None
4	Pass	The sounds play on the Login Page then the three options display after the sounds finish playing.
5	Pass	None
6	Pass	The random function on the M5 Stack doesn't appear to be random when the app is downloaded.
7	Pass	None
8	Pass	The sounds play on the Login Page then the three options display after the sounds finish playing.
9	Pass	None
10	Pass	None
11	Pass	None
12	Pass	None
13	Pass	None
14	Pass	The button options do not appear until the morse code is finished outputting
15	Pass	The button options do not appear until the morse code is finished outputting.
16	Pass	The button options do not appear until the morse code is finished outputting.
17	Pass	None
18	Pass	None
19	Pass	None
20	Pass	None
21	Pass	None
22	Pass	None
23	Pass	None

27.0 Automation Tools and Bug Tracking Tools

For the bug tracking report we used Pylint (http://www.pylint.org/). Pylint is able track and report bugs in python code. It is a tool that helps you to detect errors and potential problems in your code. In addition, Pylint can help you to detect coding style problems. For example, if you write a function that has too many lines of code, Pylint will tell you.

- Main.py bug tracking report
- Main.py GUI Command Center report

28.0 Stubs, Fakes, and Mocks

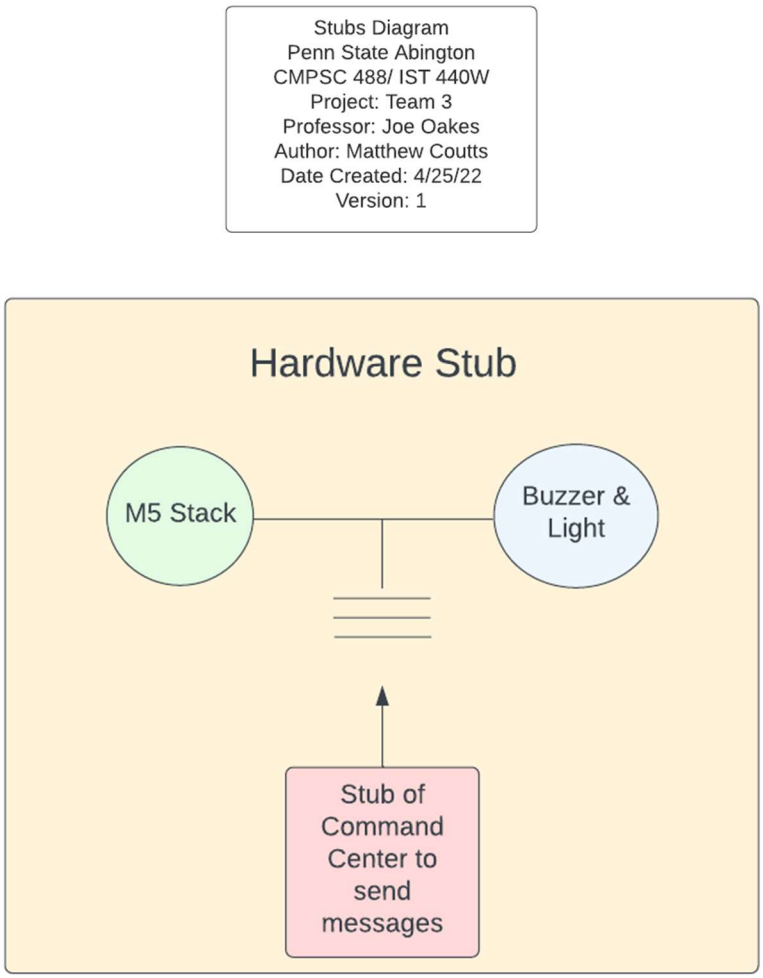


Figure 41: Send Message Stub Diagram

29.0 Project Documentation

29.1 Programmer

Escape Mission: Morse Code

Programmer Documentation

Login Page

postPuzzle(puzzleName)

parameter(s): String puzzleName return: None

Takes a puzzleName string as an input then posts that the that puzzle name has been completed to the Eve Server.

setUpLoginPage()

parameter(s): None return: None

Displays of UI elements in the Login Page.

hideLoginPage()

parameter(s): None return: None

Hides all UI elements from the Login Page.

buttonA_wasPressed()

parameter(s): None return: None

The 'default' role is selected. If a role has not been previously selected the Sound Puzzle will be setup.

buttonB_wasPressed()

parameter(s): None return: None

The 'maintenance' role is selected. If a role has not been previously selected the Flow Puzzle will be setup.

buttonC_wasPressed()

parameter(s): None return: None

The 'admin' role is selected. If a role has not been previously selected the Morse Puzzle will be setup.

Sound Puzzle

setUpSoundPuzzle()

parameter(s): None return: None

Sets up and runs the sound puzzle. Hides the login page then plays three unique tones then 1 repeated tone. The user must decide which of the three tones was repeated.

pickToneOptions()

parameter(s): None return: int[] toneOptions

Picks three unique tones from the tones array.

playTones(toneOptions)

parameter(s): int[] toneOptions return: None

Plays the tone options passed to the function.

playTonesAnswer(toneAnswer)

parameter(s): int toneAnswer return: None

Plays the tone answer passed to the function.

displayToneOptions()

parameter(s): None return: None

Displays the three tone option buttons.

tone1BtnPressed()

parameter(s): None return: None

Checks if the tone 1 is the correct answer. If it is it posts a log to the database saying the Sound Puzzle was solved. If not the Sound Puzzle is reset for another try with new tones.

tone2BtnPressed()

parameter(s): None return: None

Checks if the tone 2 is the correct answer. If it is it posts a log to the database saying the Sound Puzzle was solved. If not the Sound Puzzle is reset for another try with new tones.

tone3BtnPressed()

parameter(s): None return: None

Checks if the tone 3 is the correct answer. If it is it posts a log to the database saying the Sound Puzzle was solved. If not the Sound Puzzle is reset for another try with new tones.

Flow Puzzle

setUpFlowPuzzle()

parameter(s): None return: None

Hides the login page and displays all the buttons in the flow puzzle.

checkFlowPuzzle()

parameter(s): None return: None

Checks if every button in the puzzle is correctly set. There is only one correct configuration for the puzzle. If the puzzle is correct a log is posted to the database saying the Flow Puzzle was solved.

changeBtnColor(row, col)

parameter(s): int row, int col return: None

Changes the color of the button at the inputted coordinates.

setLastPressed(colorName)

parameter(s): String colorName return: None

Sets the current color and lastPressed variable to that of the inputted string.

redRootPressed()

parameter(s): None return: None

Sets last pressed to 'red'

orangeRootPressed()

parameter(s): None return: None

Sets last pressed to 'orange'

yellowRootPressed()

parameter(s): None return: None

Sets last pressed to 'yellow'

greenRootPressed()

parameter(s): None return: None

Sets last pressed to 'green'

blueRootPressed()

parameter(s): None return: None

Sets last pressed to 'blue'

indigoRootPressed()

parameter(s): None return: None

Sets last pressed to 'indigo'

btnReleased()

parameter(s): None return: None

When a button is released the status of notReleased is updated and the puzzle is checked if it is solved.

Morse Puzzle

initializePuzzleWords()

parameter(s): None return: None

Creates a json file called puzzleWords.json with 20 puzzle words to use in the Morse Puzzle.

setUpMorsePuzzle()

parameter(s): None return: None

Sets up and runs the Morse Puzzle. Hides the login page then picks and broadcasts a word in Morse code. The user must decode and pick the word's english translation from three options.

pickOptions()

parameter(s): None return: String[] morseOptions

Picks three unique words from puzzleWords.json

displayOptions(wordOptions)

parameter(s): String[] wordOptions return: None

Displays all the morseOption buttons.

morse1BtnPressed()

parameter(s): None return: None

Checks if the first word is the correct translation. If it is a log is posted to the database saying the Morse Puzzle was solved. If not the Morse Puzzle is reset for another try with a new word.

morse2BtnPressed()

parameter(s): None return: None

Checks if the second word is the correct translation. If it is a log is posted to the database saying the Morse Puzzle was solved. If not the Morse Puzzle is reset for another try with a new word.

morse3BtnPressed()

parameter(s): None return: None

Checks if the third word is the correct translation. If it is a log is posted to the database saying the Morse Puzzle was solved. If not the Morse Puzzle is reset for another try with a new word.

Broadcast Message

initializeLetterTranslations()

parameter(s): None return: None

Creates a json file called characterConversions.json with the translation with each letter's translation in morse code.

translateString(string)

parameter(s): String strings return: String output

Takes a string input and returns the morse code translation as a string with '.' '-' and '/' symbols.

broadcastMessage(message)

parameter(s): String message return: None

Broadcasts an english string message to the LED and Speaker sensors.

dot()

parameter(s): None return: None

Outputs the dot symbol on the LED and Speaker sensors.

dash()

parameter(s): None return: None

Outputs the dash symbol on the LED and Speaker sensors.

letterSpace()

parameter(s): None return: None

Waits for the space between letters.

slash()

parameter(s): None return: None

Waits for the space between words.

sameLetterSpace()

parameter(s): None return: None

Waits for the space between symbols in the same letter.

GUI Application

readMessages(host, port)

parameter(s): String host, String port return: None

Connects to the Eve Server using the host and port parameters. Gets all the messages stored in the mongoDB messages collection and returns it as a dictionary.

saveMessage(host, port, message, numberOfMessages=1)

parameter(s): String host, String port, String message, int numberOfMessages return: None

Connects to the Eve Server using the host and port parameters. Creates a new message entry with the message parameter and the numberOfMessages parameter.

deleteMessage(host, port, message_id)

parameter(s): String host, String port, String message_id return: None

Connects to the Eve Server using the host and port parameters. Deletes the message from the database with the message_id parameter.

29.2 User

Escape Mission: Morse Code

User Documentation

Mission Statement

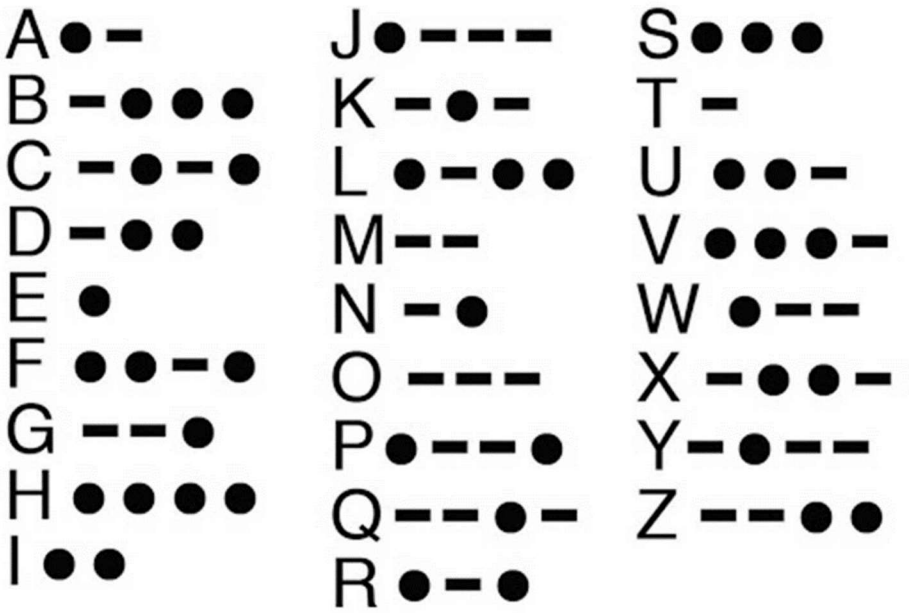
We will be making a communication system utilizing Morse code with escape room elements. The purpose of the application is to establish and conduct communication with mission control back on Earth to coordinate a rescue.

The application will allow users to input messages in English that will be broadcast into space. These messages will be translated to Morse code in the application then sent to the M5 stack to be broadcast.

The Morse code messages will be broadcast into space via the LED and Buzzer Module through the M5 stack using Morse code conventions.

To use the communicator, the user will have to complete a puzzle where a word will be displayed in morse code, and they must translate it (with the use of a guide sheet) and choose the correct word from the options provided.

Morse Code



Morse code is a method of communication where each character is encoded into a series of dot and dash symbols as shown above. A **dot** symbol is a sound or light for **1 unit of time**. A **dash** symbol is represented by a sound and/or light for **3 units of time**. A **space between symbols** within a letter is represented by no light and/or sound for **1 unit of time**. A **space between letters** is represented by no light and/or sound for **3 units of time**. A **space between words** is represented by no light and/or sound for **7 units of time**.

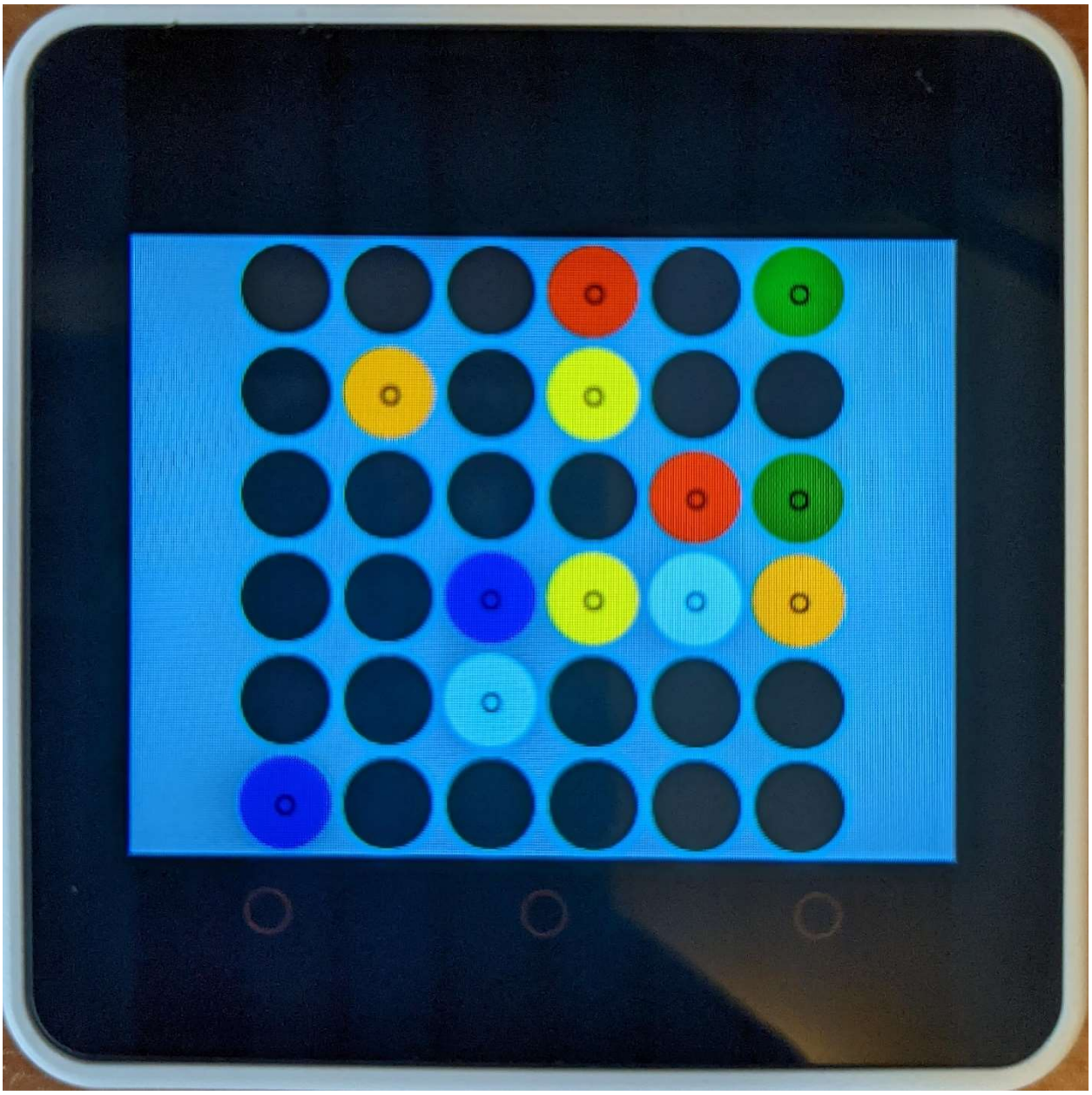
Puzzles

Sound Puzzle

The Sound Puzzle consists of playing three separate sound then repeating one of the sounds. The goal of the puzzle is to determine which of the three sounds was repeated.

The Sound Puzzle is used to **login** as a astronaut or **default** user which has the lowest level of access. The astronaut or default user only has access to **send messages** and **message loops**.

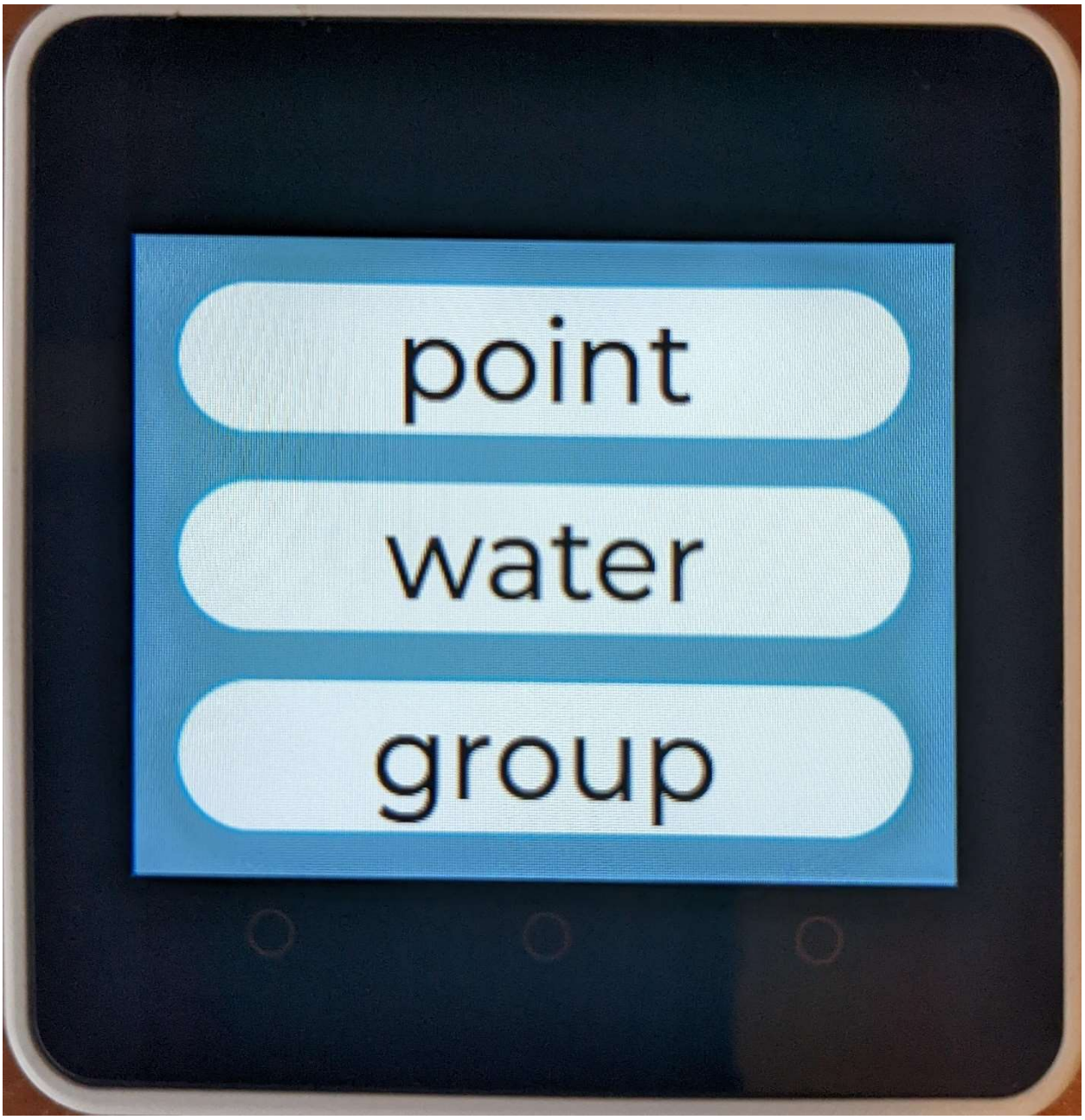
Flow Puzzle



The Flow Puzzle is starts as a screen with two nodes of each color. To complete the puzzle the user must cover the whole screen and connect the root nodes of each color with paths without branching.

The Flow Puzzle is used to **login** as a **maintenance** user. The maintenance user has access to **send messages** and **message loops** as well as use the **test translation** page. On the test translation page the maintenance user can input English and/or morse code and see the translation. This can be used to determine what the translation will outputed as. It also can be used to check the translation works correctly.

Morse Puzzle

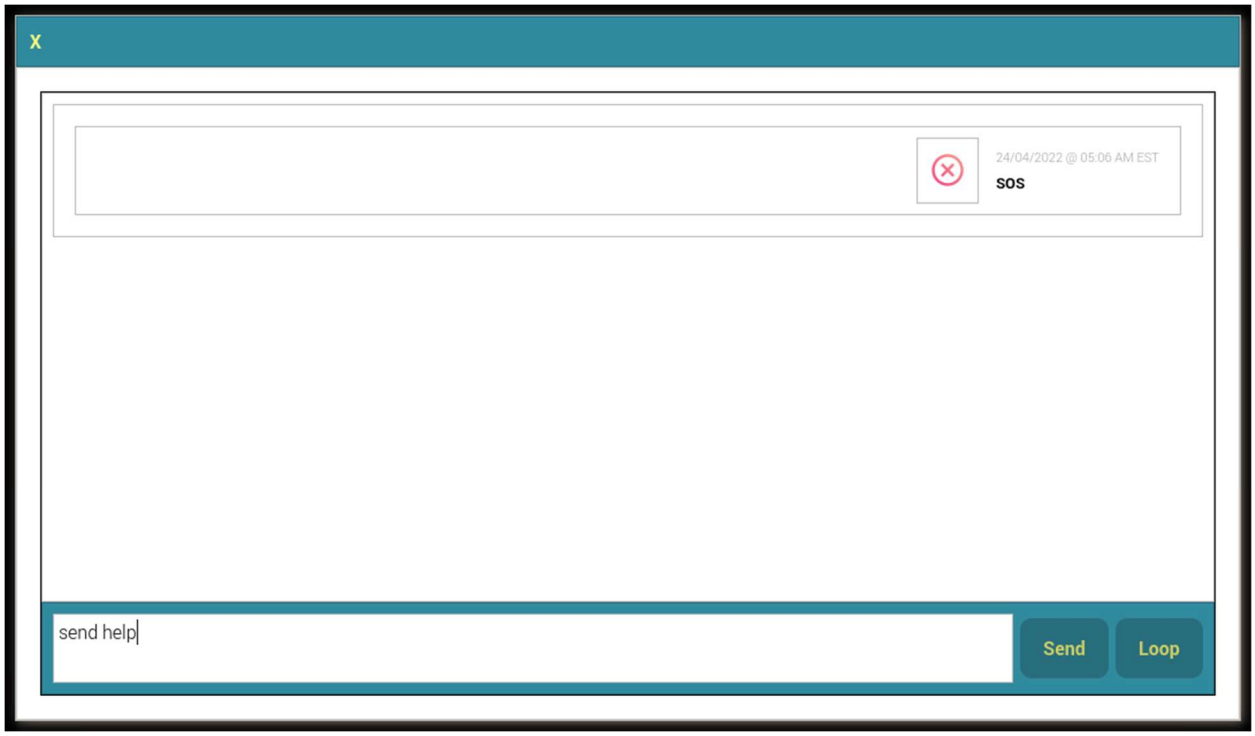


The Morse Puzzle will out put a random word in morse code. The goal of the puzzle is to pick the translation of the word into English from three provided options.

The Morse Puzzle is used to **login** as an **admin** user. The admin user has access to **send messages** and **message loops** as well as use the **test translation** page. The admin user can also **delete** previously sent messages.

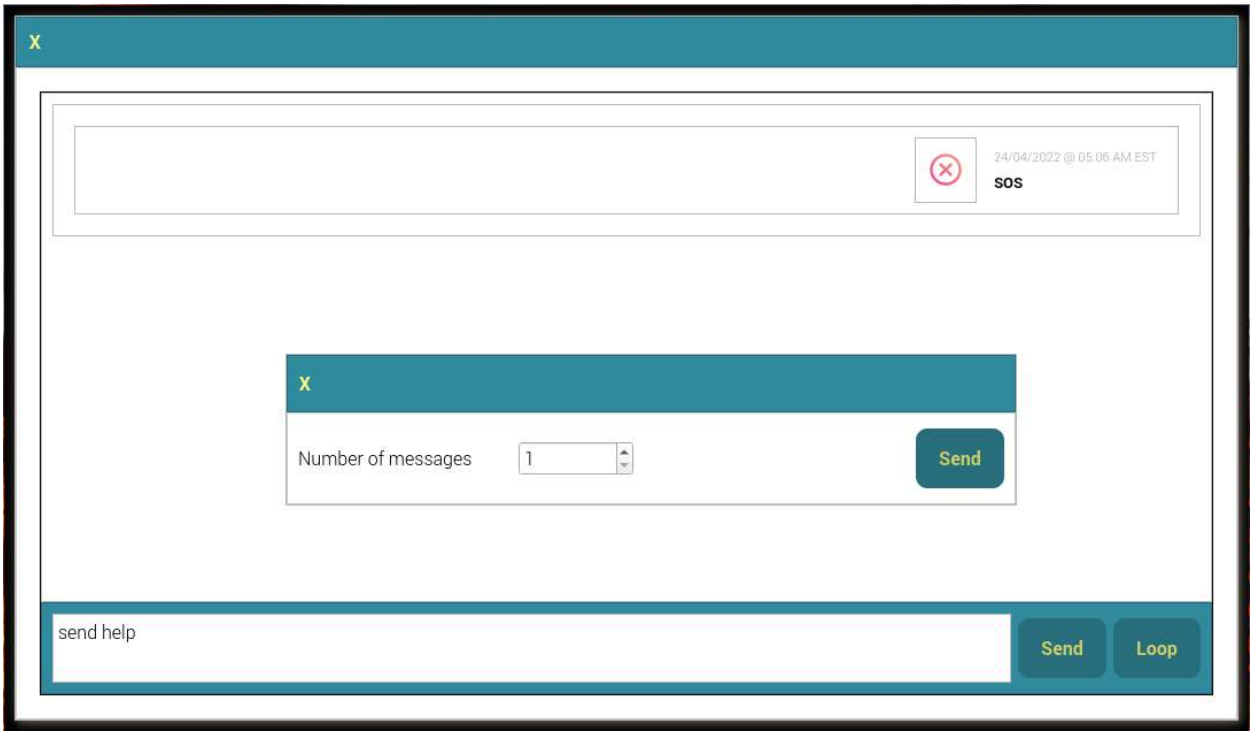
GUI Application

Send Message



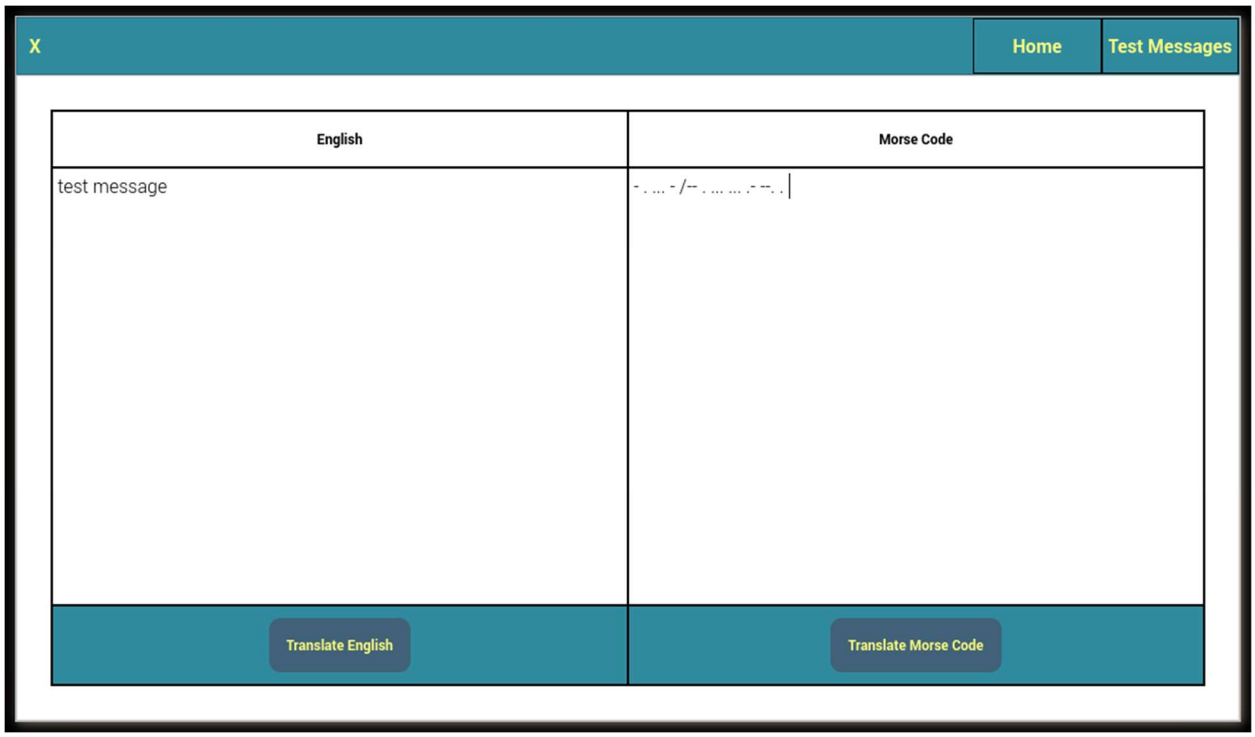
To send a message the user can type a message using only characters from the English alphabet and click the send button. The message will then be sent to the M5 Stack where it will then be broadcast via the sensors on the M5 Stack.

Send Message Loops



Similar to sending a message the user has the option to send a message loop. To send a message loop the user can type a message using only characters from the English alphabet and click the loop button. The user will then be prompted to enter how many times the message should loop. The message will then be sent to the M5 Stack where it will be broadcast the number times specified in the loop pop up.

Test Translation



In the test translation page allows the maintenance and admin user to view what an English word or phrase will look like in Morse Code or vice versa. To use the page input

29.3 Admin

Escape Mission: Morse Code

Admin Documentation

Application Setup

M5 Stack

If the app is not installed on the M5 Stack follow the following steps to install the app:

1. Go to the website https://flow.m5stack.com/
2. Connect M5 Stack to M5Flow
3. Click "Python" button at the top of the website
4. Copy and paste the Main.py file into M5Flow
5. Enter the Name and Password (line 26) for your WiFi
6. Change the host (line 38) to IP address of your WiFi
7. Click the "Download" button in the lower right-hand corner

After completing these steps the M5 Stack should restart and the Login Page will load.

Raspberry Pi

To start the application on the Raspberry Pi follow the steps below: Download the deployEve.sh and deployGUI.sh (Follow the link to each script right-click on the page and click "Save Page As...")

1. In the command line change directory to where the deployment scripts are saved (cd *Script/Directory/Path*)
2. Run the command below to deploy the Eve Server
   `bash deployEve.sh`
3. Run the command below to start the GUI Application
   `bash deployGUI.sh`

The GUI application should start. You can then login to the GUI Application by completing a puzzle on the M5 Stack then start sending messages.

30.0 Deployment Scripts

30.1 Eve Deployment Script

```
# MongoDB Setup
echo "Updating operating system..."
sudo apt update
sudo apt upgrade

echo ""
echo "Downloading MongoDB..."
wget -qO - https://www.mongodb.org/static/pgp/server-4.4.asc | sudo apt-key add -
# Install the MongoDB 4.4 GPG key

echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu focal/mongodb-
org/4.4 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-4.4.list # Add
the source location for the MongoDB packages
sudo apt-get
update                                                          # Download
the package details for the MongoDB packages
sudo apt-get install -y mongodb-
org                                             # Install MongoDB

echo ""
echo "Starting MongoDB..."
sudo systemctl daemon-
reload                                                  # Ensure mongod
config is picked up
sudo systemctl enable
mongod                                                  # Tell systemd to run
mongod on reboot
```

```
sudo systemctl start
mongod                                                  # Start up mongod


# Eve Server Setup
cd ~/Documents
if [ -d "CommandCenter" ]; then rm -Rf "CommandCenter"; fi

echo ""
echo "Cloning CommandCenter repository..."
git clone
https://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@github.com/qqm5052/IST440CC.git
CommandCenter

echo ""
echo "Installing dependencies..."
cd CommandCenter
pip install -r requirements.txt

echo ""
echo "Starting Eve server..."
python3 starteve.py 192.168.1.234 5000
```

30.2 GUI Application Deployment Script

```
cd ~/Documents
if [ -d "Team3Project" ]; then rm -Rf "Team3Project"; fi

echo "Cloning Team3Project repository..."
git clone
https://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@github.com/joeoakes/abist440sp22
Team3.git Team3Project

echo ""
echo "Installing dependencies..."
cd Team3Project
pip install -r requirements.txt

echo ""
echo "Running GUI application..."
cd Project/GUI
python3 main.py 192.168.1.234 5000
```

Document Change Revisions

Version	Change Date	Reason	Change Area	Author
1.0	4/10/2022	Document Formatting	Entire Document	Matt
1.1	4/24/2022	Added Deployment Scripts	Deployment Scripts	Bradley
1.2	4/24/2022	Added Project Documentation	Project Documentation	Almaz
1.3	4/25/2022	Added Test Plan	Test Plan	Bradley
1.4	4/25/2022	Added Test Plan	Test Plan	Almaz
1.5	4/29/2022	Updated User Documentation	Project Documentation	Bradley
1.6	4/29/2022	Updated Admin Documentation	Project Documentation	Bradley
1.7	4/29/2022	Added Programmer Documentation	Project Documentation	Bradley
1.8	4/29/2022	Added Incidents Report	Testing Incidents and Summary Reports	Bradley

Glossary of Terms

Breadboard - Prototyping board used for electronics. (Breadboard)

LED (Light Emitting Diode) - Semiconductor light with low power consumption. (Light-emitting diode)

Morse Code - Communication system using a series of ‘dots’ (short flashes), ‘dashes’ (long flashes), and spaces in between. (Morse code)

M5 Stack - M5Stack is a robust, open-source development kit with stackable modules, user-friendly IDE, enabling rapid and high-quality prototyping. (Project | M5Stack - An Open Source Enclosed Modular Toolkit | Hackaday.io)

Raspberry Pi - a low-cost, small computer that has many capabilities (What is a Raspberry Pi?)

UpperCamelCase - a practice of writing phrases without spaces or punctuation where the first letter in each word is capitalized.

PascalCase - a practice of writing phrases without spaces or punctuation where the first letter in each word is capitalized.

lowerCamelCase - a practice of writing phrases without spaces or punctuation where the first letter in the phrase is lower case then the first letter in each new word is capitalized.

Works Cited

"17 User story examples for when the ink runs dry." n.d. *Just In Mind.* Article. 23 01 2022. <https://www.justinmind.com/blog/user-story-examples/>.

"Breadboard." 15 11 2021. *Wikipedia.* 23 01 2022. <https://en.wikipedia.org/wiki/Breadboard>.

"Light-emitting diode." 22 01 2022. *Wikipedia.* 23 01 2022. <https://en.wikipedia.org/wiki/Light-emitting_diode>.

"Morse code." 13 01 2022. *Wikipedia.* 23 01 2022. <https://en.wikipedia.org/wiki/Morse_code>.

"Project | M5Stack - An Open Source Enclosed Modular Toolkit | Hackaday.io." n.d. *Hackaday.* Article. 23 01 2022. <hackaday.io/project/14524/logs?sort=oldest>.

"Resistor." 20 01 2022. *Wikipedia.* 23 01 2023. <https://en.wikipedia.org/wiki/Resistor>.

"What is a Raspberry Pi?" 20 08 2015. *Raspberry Pi.* 23 01 2022. <https://www.raspberrypi.org/help/what-%20is-a-raspberry-pi/>.

“Color Palette: #1A374D #406882 #6998AB #B1D0E0 - Color Hunt.” Color Hunt, colorhunt.co/palette/1a374d4068826998abb1d0e0. Accessed 6 Feb. 2022.

“Color Palette: #FFF89A #FFC900 #086E7D #1A5F7A - Color Hunt.” Color Hunt, colorhunt.co/palette/fff89affc900086e7d1a5f7a. Accessed 6 Feb. 2022.

“Flow.” *XP Game Plus - Play Free Online Games*, Apps Studio Inc., https://www.xpgameplus.com/games/flowfree/index.html.

“Play Flow Lines.” Lagged, lagged.com/en/g/flow-lines. Accessed 6 Feb. 2022.

Dwivedi, Neelam. "Use case diagram - Software Design: Modeling with UML." 23 09 2019. *LinkedIn.* Video. 23 01 2022. <https://www.linkedin.com/learning/software-design-modeling-with-uml/use-case-diagram?u=76811570>.

Dwivedi, Neelam. "Use-cases - Software Design: Developing Effective Requirements." 18 09 2019. *LinkedIn.* Video. 23 01 2022. <https://www.linkedin.com/learning/software-design-developing-effective-requirements/use-cases?autoAdvance=true&autoSkip=false&autoplay=true&resume=true&u=76811570>.

GitHub. n.d. 3 April 2003. <https://github.com/>.

Penn State University. n.d. 23 01 2022. <https://www.psu.edu/>.

Popat , Vivek. “Free UI Style Guide for Adobe XD.” *XDGuru.com*, 30 Mar. 2017, https://www.xdguru.com/free-ui-style-guide/.

Stone, Barron. "Use cases - Python Projects." 24 09 2021. *LinkedIn.* Video. 23 01 2022. <https://www.linkedin.com/learning/python-projects-14276284/use-cases?autoAdvance=true&autoSkip=false&autoplay=true&resume=true&u=76811570>.

“Welcome to Bandit¶.” *Welcome to Bandit - Bandit Documentation*, Bandit Developers Revision, https://bandit.readthedocs.io/en/latest/.

Wick, Angela. "Data flow diagrams - Requirements Elicitation and Analysis." 20 08 2017. *LinkedIn.* Video. 30 01 2022. <https://www.linkedin.com/learning/requirements-elicitation-and-analysis/data-flow-diagrams-8360499?trk=learning-serp_learning-search-card_search-card&upsellOrderOrigin=default_guest_learning>.

Team Signatures

Almazbek Akhunbaev: Almazbek Akhunbaev

Alexander Djordjevic: Alexander Djordjevic

Matthew Coutts: Matthew Coutts

Bradley Whitebread: Bradley Whitebread

Md Maruf Mollah: Md Mollah