GNG<1103/2101>

Design Project User and Product Manual

Submitted by:

Zonify Group 16.

Dennis Baik, 300404910

Tariq Mustapha, 300421081

Taryn McDermid, 300410820

Rachael Neff, 300425179

William Gillespie, 300444324

11/28/24

University of Ottawa

List of Figures

List of Tables

Bandwidth	The amount of information that can be sent through the network
API	A connection between applications to send and receive requested information
UI	User Interface

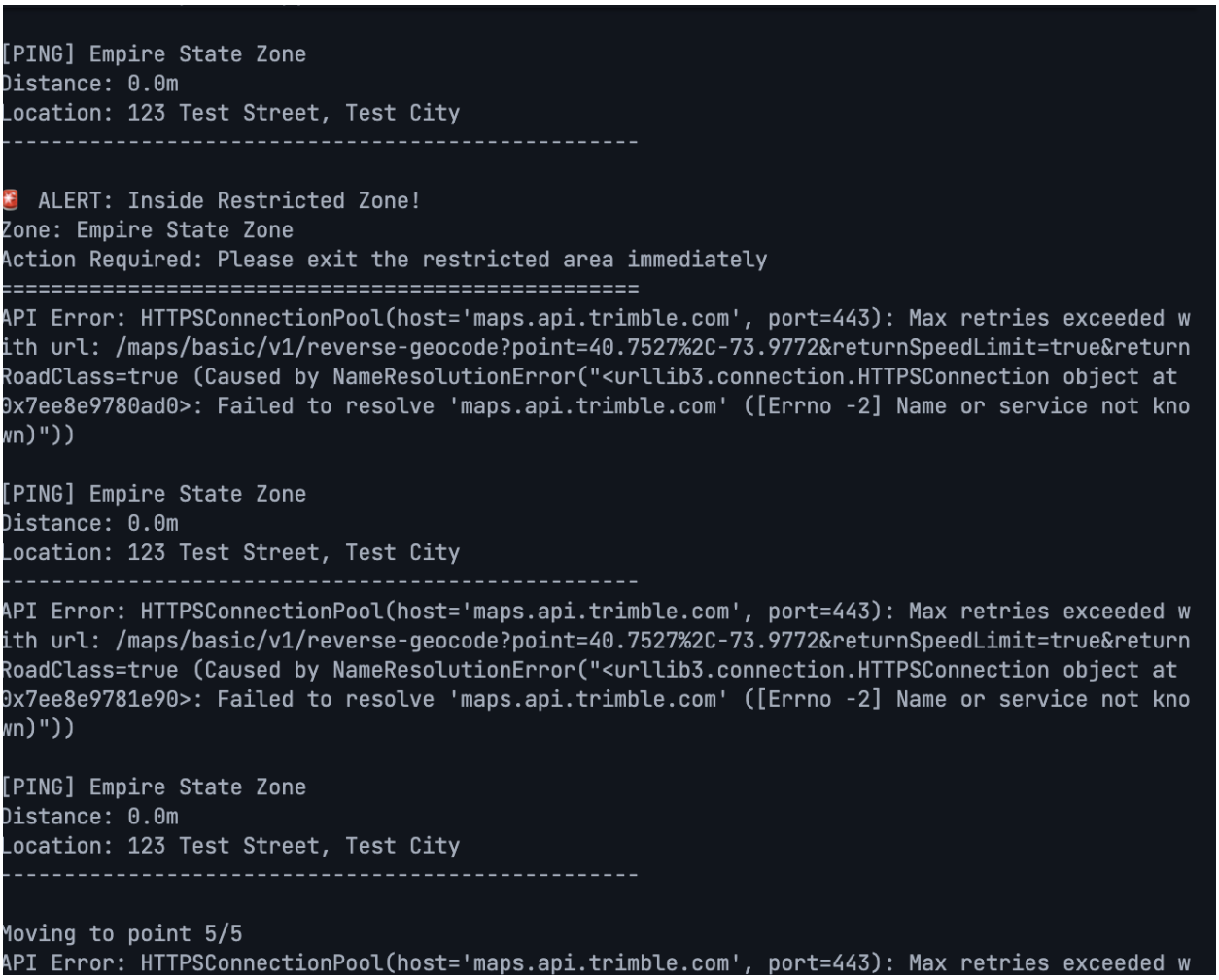
1 Introduction

This User and Product Manual (UPM) provides the information necessary for Developers and users to effectively use Zonify and for prototype documentation. It will highlight any warnings, usage, troubleshooting and product documentation.

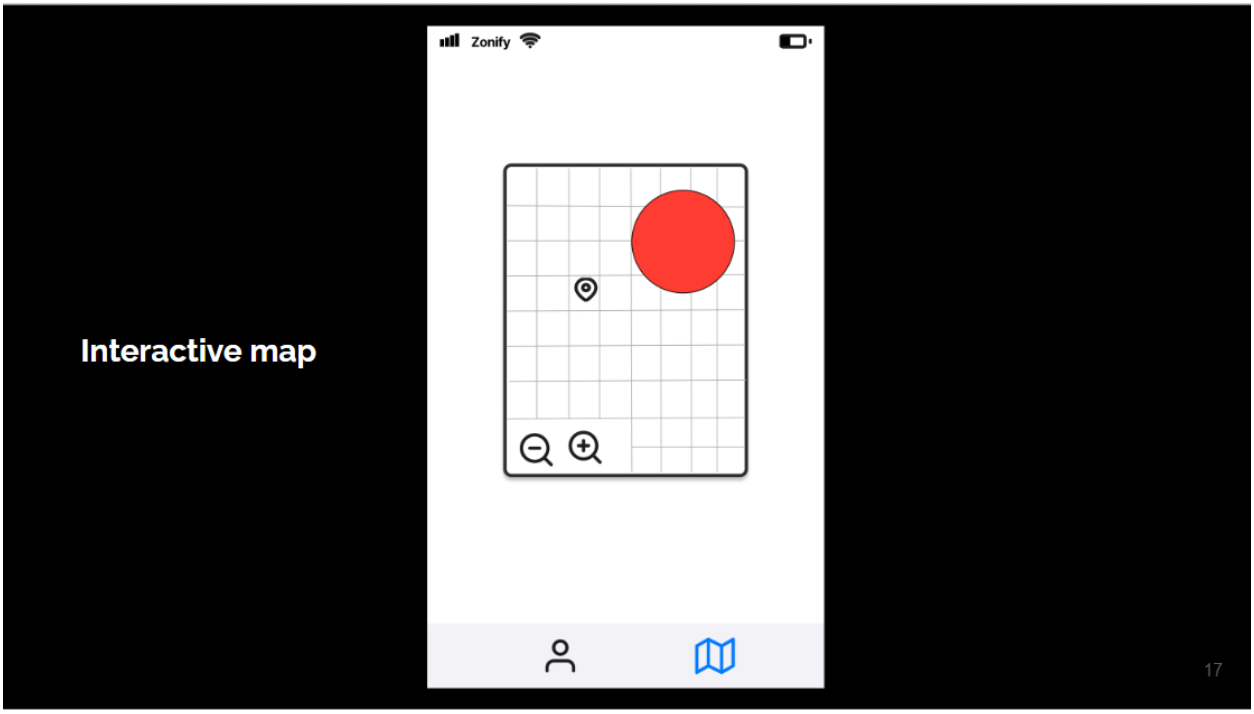
2 Overview

The company Shabodi required a network aware, modular code that will alert administration and the user of the device that they are approaching a restricted zone. It used at least 2 API's and is capable of demonstration in their sandbox.

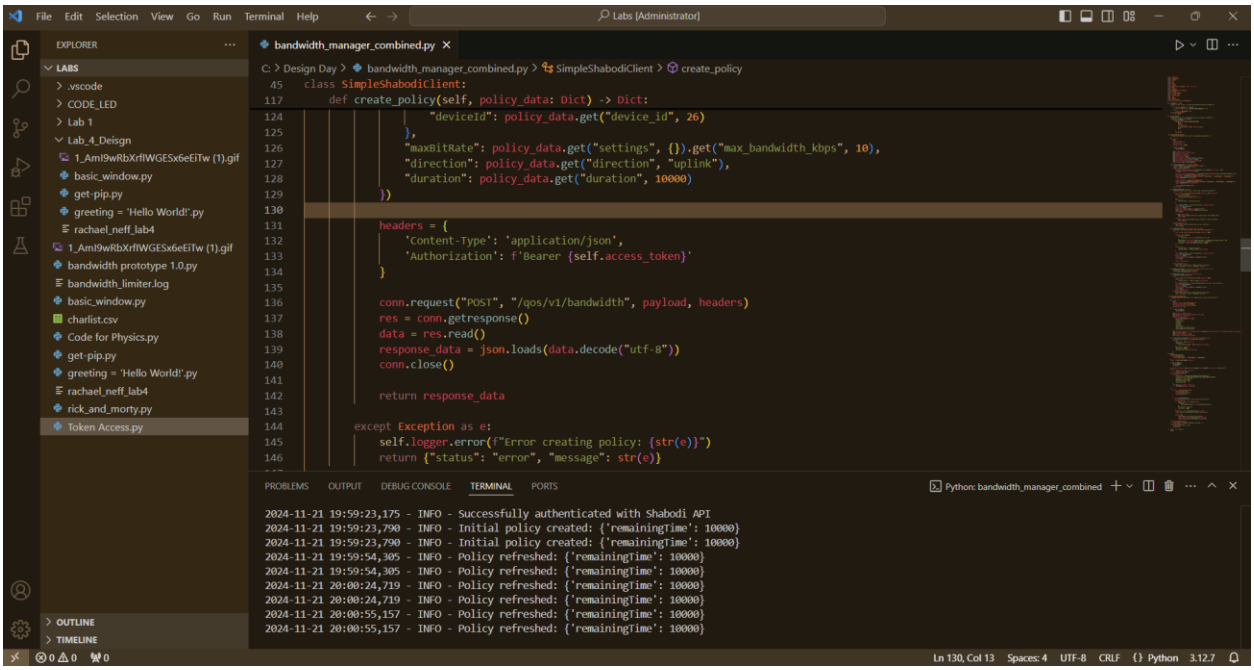
Pictures of Prototype:



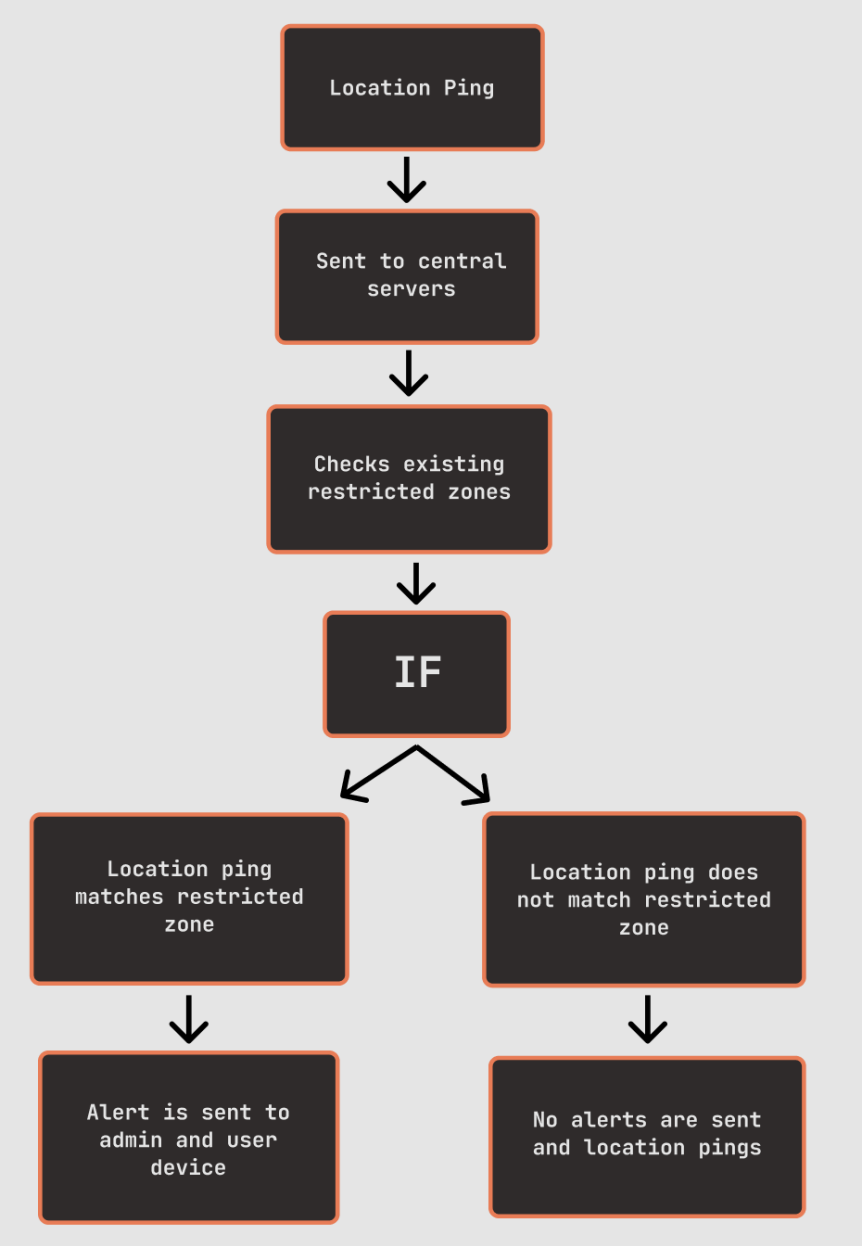
The location code shown above demonstrates the code in how it checks the current device location and sends alerts when required.



The UI above shows an example of the map shown to users, the red being the zone of restriction.



The above bandwidth code, when prompted by the location code creates limits on the device bandwidth.



The block diagram shows the logical steps the program will take to check and limit the bandwidth depending on the location.

2.2 Cautions & Warnings

Network Connectivity: Requires a stable internet connection to function properly. Ensure that you have a reliable Wi-Fi or mobile data connection to avoid interruptions in service

Shabodi VPN Connection: A VPN connection to shabodi is required to access their API's.

Compatibility: Ensure that your device meets the minimum system requirements for the app.

Waiver Use or Copy Permissions:

Permissions from Shabodi API services are required to access the API responses.

3 Getting started

The majority of our software and code was created using generative AI, this allowed our team to create the best possible product using our limited coding experience. As of right now our system is split up into three different coding sequences, the code for the location API, the code for the bandwidth API, and the code for the app software. Below there is a detailed walk through of each of these programs.

Location API Code:

The location API code uses 3 different pips and a google geolocation API was used in place of Shabodi, due to issues with their Location API services. The API software was then integrated into a zone restriction code not only to monitor where a person was but also to alert someone when they were in a restricted zone. This code utilized a single longitude and latitude point along with the radius of a circle in order to create a restricted zone.

Bandwidth API Code:

Once prompted by the location code, the bandwidth code will begin to limit the device that has been identified to have entered the restricted zone. Once the device has left the restricted zone the code will stop restricting the bandwidth.

Application Code:

The application primarily utilizes buttons and text boxes for user interaction, which can be easily implemented using a basic graphical user interface (GUI) in Python by leveraging libraries such as Tkinter or PyQt.

3.1 Configuration Considerations

For software prototypes: Briefly describe and graphically depict as appropriate the communications and configuration of the system in a way that a non-technical user can understand. Include the type of input and output devices/databases needed.

Input:

- Need device (smartphone, tablets computers)
- access to 5g network
- Stores user-provided information

Output

- Contain information that is displayed to users
- Need device (smartphone, tablets computers)

3.2 User Access Considerations

Describe the different users and/or user groups that could be using the prototype, and the restrictions placed on accessibility or use for each.

User:

- Accessibility: Little to no control over the application itself. When using the app the only things that they can see is their location in context to a restricted zone and they will receive notifications when they are approaching and in a restricted zone.
- Restrictions: A user will only see the restricted zones that apply to them and will not be able to see the location of others.

Administrators:

- Accessibility: Will have control over whether or not the bandwidth throttling code runs when a person is in a restricted zone. They will also be able to set and change restricted zones. Additionally, the administrator and any other employee that is set to receive notifications (such as security personnel) will receive a notification any time a person enters a restricted zone who is not supposed to be there.
- Restrictions: Administrators will not be able to see where their employees are at all times, they will only be able to see when they are in a restricted zone.

3.3 Accessing/setting up the System

Location Prototype:

To access this code, you will need a link to the IDX platform on which this code was written. After accessing this you may need to regenerate the google geolocation API key and insert that into the corresponding spot within the code. If this doesn't work then copy the code and copy it into IDX. Then

Bandwidth Prototype:

To access this code, you will need to have access to Shabodi's VPN and be able to generate a key. Once you are able to do this copy the code into visual studios code and run the code in administrator mode.

Application Prototype:

To access this export, it into visual studio codes.

3.4 Exiting the System

To exit the system, simply terminate the program at the current IDE.

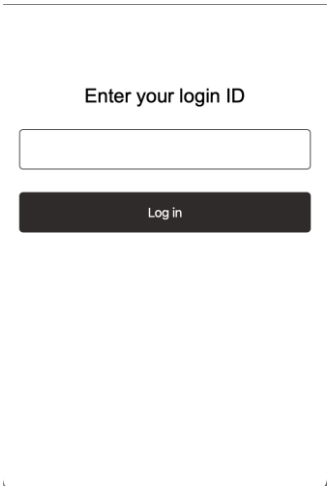
4 Using the System

Using the system as a whole is intuitive and simple. Administrators can create a restricted zone, control bandwidth of devices inside the restricted zone, and track location of devices in relation to the restricted zone. Devices with user access have limited functionality and can only access a map that tracks their device, and the precise location of the restricted zone set by the administrator.

The following sub-sections provide detailed, step-by-step instructions on how to use the various functions of the application.

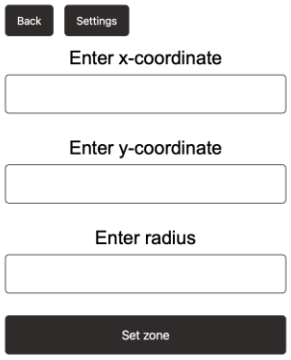
4.1 Log In Menu

To log in to the application, a certain code should be provided to the device that will allow the device to run the application. Each code is unique to the device and certain codes allow devices access to administrator functionality where they can create restricted zones and toggle bandwidth.



4.2 Create a Restricted zone

To enter a restricted zone, the administrator must enter the precise longitude (x – axis) and latitude (y – axis) coordinates and the radius of the circle that will surround the coordinates. The area in this circle will be entered into the system and any unauthorized devices in this area will be alerted.



4.3 Controlling Bandwidth

Administrators can choose to choke the bandwidth of unauthorized devices when they enter the restricted zone by going into the settings page and turning the bandwidth button OFF.

The other option is to limit the bandwidth by only allowing essential functions of the device to continue, such as emergency calls and location pinging. To do this, the bandwidth toggle must be switched to ON.

This feature allows the application to be useful in different sit

5 Troubleshooting & Support

VPN Connection Error: The VPN will disconnect from this application every 3 hours and needs to be reconnected to, to allow application to run with shabodis API's.

Shabodi API Connection Error: Shabodi's API's may be down or unable to connect to. The connection error can indicate your token generation credentials. Please contact Shabodi developer.support@shabodi.com for further information.

Application Running Errors: Depending on what application you are running, it may be incompatible with the software Tkinter. In such cases, either run the application on a different hardware or software environment to successfully run the code.

5.1 Error Messages and Behaviours

5.1.1 VPN Connection Error

You may encounter an error indicating that the connection to the API has failed, or the application will stop working because the VPN is disconnected.

5.1.2 Shabodi API Connection Error

The application may show a message indicating that the API token cannot be verified, or there might be a failure to establish a connection to the API.

5.1.3 Application Running Error

The application may fail to launch or may show an error code.

5.2 Maintenance

If the current ID and secret key to access Shabodi's API are no longer up to date and token generation is unavailable, please contact Shabodi developer.support@shabodi.com to update your credentials.

5.3 Support

If Shabodi's API or VPN is unavailable or being worked on. Contact Shabodis at developer.support@shabodi.com. For further support email customer-support@shabodi.com or call +1- 289-800-5990.

6 Product Documentation

The final prototype was made with 3 separate parts, the location service code, the bandwidth restriction code and the user interface code. While it was not possible to integrate all three parts of the code into one product, with adequate time investment it would be possible to result in a single complete product.

For the location service code, a public API named Trimble was initially used as proof of concept and later integrated into Google API as the Shabodi location API was not functional. Multiple iterations with Claude AI were done with the prompts being:

1. I need you to create a code that uses a location API. This code will be used to create restricted zones, when a restricted zone is created the code needs to be able to alert a user not only when they are approaching a restricted zone but also when they are in one. There should be two different sounds and vibrations for those notifications. We should also be able to change and customize a restricted zone
2. complete this code in python.
3. can you add in a pinging section, where it will ping your location the closer you are to the restricted zone and it will ping it less the further you are away from the restricted zone
4. write it using this API key **47C4ECCCB1EF134EB784BDB1660FD90B**
5. It is for trimble maps and this is a publically available API
6. can you fill in some random info in the blank parts so that I can test the code
7. can you take the code you have already given me but take out the pygame and the notification portion because that has been causing me issues\
8. Now can you add a notification system that will notify a person when they are approaching and when they are in a restricted zone, can you try not to use pygame because i couldn't get that to work on my computer
9. in line 53 i am getting an invalid syntax error
10. it says no module names modify two
11. can you replace all of the places with tremble mentioned with a place to input our own api information
12. the api i am going to use is a location api so I do not have a different api maps key

13. Can you change this code so that it is set up to utilize googles geolocation api:
14. where do I incert googles api key
15. can you highlight where that is in my code
16. i dont see that in my code can you add it
17. idx says that no googlemaps module found
18. how can i connect my idx code with my google cloud account so i can use the api
19. will this allow me to use the google api in my idx code
20. i am getting this error API Error: REQUEST_DENIED (This API project is not authorized to use this API.)
21. i am not able to select my idx project though i can not find it
22. can you ensure that the test section of my code does no need the use of a speed api
23. Yes

The code created with these prompts is as follows:

```
import math
import time
from dataclasses import dataclass
from typing import Dict, Optional, List
from threading import Thread
import googlemaps
from dotenv import load_dotenv
import os

# Load environment variables
load_dotenv()

# Initialize Google Maps client
API_KEY = os.getenv('GOOGLE_MAPS_API_KEY')

@dataclass
class Point:
lat: float
lng: float

class GoogleMapsAPI:
def __init__(self, api_key: str):
self.client = googlemaps.Client(key="AIzaSyA2gvK4YCW3g7TLilSOPwVtxAPzpqOKcFM")

def reverse_geocode(self, point: Point) -> dict:
"""Get location information from coordinates."""
try:
result = self.client.reverse_geocode((point.lat, point.lng))
```

```
if result:
address = result[0]
return {
"address": {
"freeformAddress": address['formatted_address'],
"country": next((c['long_name'] for c in address['address_components']
if 'country' in c['types']), ''),
"countryCode": next((c['short_name'] for c in address['address_components']
if 'country' in c['types']), '')
}
}
return self._get_dummy_data(point)
except Exception as e:
print(f"API Error: {e}")
return self._get_dummy_data(point)

def get_route(self, start: Point, end: Point) -> dict:
"""Get routing information between two points."""
try:
directions = self.client.directions(
origin=(start.lat, start.lng),
destination=(end.lat, end.lng),
mode="driving"
)
if directions:
route = directions[0]
path = self._decode_polyline(route['overview_polyline']['points'])
return {
"geometry": {
"type": "LineString",
"coordinates": [[p[1], p[0]] for p in path]
},
"summary": {
"distance": route['legs'][0]['distance']['value'],
"time": route['legs'][0]['duration']['value']
}
}
return self._get_dummy_route(start, end)
except Exception as e:
print(f"API Error: {e}")
return self._get_dummy_route(start, end)

def _decode_polyline(self, polyline: str) -> List[tuple]:
"""Decode Google's polyline format."""
points = []
index = lat = lng = 0
```

```
while index < len(polyline):
result = 1
shift = 0
while True:
b = ord(polyline[index]) - 63 - 1
index += 1
result += b << shift
shift += 5
if b < 0x1f:
break
lat += (~result >> 1) if (result & 1) else (result >> 1)

result = 1
shift = 0
while True:
b = ord(polyline[index]) - 63 - 1
index += 1
result += b << shift
shift += 5
if b < 0x1f:
break
lng += (~result >> 1) if (result & 1) else (result >> 1)

points.append([lat * 1e-5, lng * 1e-5])

return points

def _get_dummy_data(self, point: Point) -> dict:
"""Return dummy data for testing."""
return {
"address": {
"freeformAddress": "123 Test Street, Test City",
"country": "United States",
"countryCode": "US"
}
}

def _get_dummy_route(self, start: Point, end: Point) -> dict:
"""Return dummy route data for testing."""
return {
"geometry": {
"type": "LineString",
"coordinates": [
```

```
[start.lng, start.lat],
[start.lng + 0.001, start.lat + 0.001],
[end.lng, end.lat]
]
},
"summary": {
"distance": 1000,
"time": 300
}
}

class RestrictedZone:
def __init__(self, zone_id: str, center: Point, radius: float, name: str = ''):
self.id = zone_id
self.center = center
self.radius = radius # in meters
self.name = name
# Define distance thresholds for ping frequency (in meters)
self.distance_thresholds = {
100: 5, # Within 100m: ping every 5 seconds
250: 60, # Within 250m: ping every 60 seconds
500: 600, # Within 500m: ping every 10 minutes
1000: 900, # Within 1km: ping every 15 minutes
2000: 1800, # Within 2km: ping every 30 minutes
float('inf'): 1800 # Beyond 2km: ping every 30 minutes
}
# Store zone metadata
self.location_info = None

def update_location_info(self, google_api: GoogleMapsAPI):
"""Update zone location information using Google Maps API."""
try:
location_data = google_api.reverse_geocode(self.center)
self.location_info = location_data
except Exception as e:
print(f"Error updating location info: {e}")

def is_inside(self, point: Point) -> bool:
return self.get_distance(point) <= self.radius

def is_approaching(self, point: Point) -> bool:
distance = self.get_distance(point)
return self.radius < distance <= (self.radius + 100)

def get_distance(self, point: Point) -> float:
```

```
R = 6371e3 # Earth's radius in meters
φ1 = math.radians(self.center.lat)
φ2 = math.radians(point.lat)
Δφ = math.radians(point.lat - self.center.lat)
Δλ = math.radians(point.lng - self.center.lng)

a = (math.sin(Δφ/2) * math.sin(Δφ/2) +
math.cos(φ1) * math.cos(φ2) *
math.sin(Δλ/2) * math.sin(Δλ/2))
c = 2 * math.atan2(math.sqrt(a), math.sqrt(1-a))

return R * c

def get_ping_interval(self, distance: float) -> float:
for threshold, interval in sorted(self.distance_thresholds.items()):
if distance <= threshold:
return interval
return self.distance_thresholds[float('inf')]

class GeofencingSystem:
def __init__(self, api_key: str):
self.google_api = GoogleMapsAPI(api_key)
self.zones: Dict[str, RestrictedZone] = {}
self.watching = False
self.last_alerts = {}
self.last_pings = {}
self.alert_cooldown = 30
self.current_position = None

def add_zone(self, center: Point, radius: float, name: str = '') -> str:
"""Add a new restricted zone."""
zone_id = str(time.time())
zone = RestrictedZone(zone_id, center, radius, name)
# Get location information
zone.update_location_info(self.google_api)
self.zones[zone_id] = zone
self.last_pings[zone_id] = 0
return zone_id

def get_current_location(self) -> Optional[Point]:
"""Get current location."""
return self.current_position

def simulate_movement(self, points: List[Point]):
```

```
"""Simulate movement along a path of points."""
if not points or len(points) < 2:
return

for i in range(len(points) - 1):
try:
route = self.google_api.get_route(points[i], points[i + 1])
if 'geometry' in route:
for coord in route['geometry']['coordinates']:
self.current_position = Point(lat=coord[1], lng=coord[0])
self._check_zones(self.current_position)
time.sleep(1)
except Exception as e:
print(f"Error simulating movement: {e}")

def _check_zones(self, current_point: Point):
"""Check all zones against current position and handle pinging."""
current_time = time.time()
for zone in self.zones.values():
distance = zone.get_distance(current_point)
ping_interval = zone.get_ping_interval(distance)
last_ping = self.last_pings.get(zone.id, 0)
last_alert = self.last_alerts.get(zone.id, 0)

if current_time - last_ping >= ping_interval:
self._ping_location(zone, distance)
self.last_pings[zone.id] = current_time

if current_time - last_alert > self.alert_cooldown:
if zone.is_inside(current_point):
self._alert_inside_zone(zone)
self.last_alerts[zone.id] = current_time
elif zone.is_approaching(current_point):
self._alert_approaching_zone(zone)
self.last_alerts[zone.id] = current_time

def _ping_location(self, zone: RestrictedZone, distance: float):
"""Console output for location pings."""
if self.current_position:
try:
location_info = self.google_api.reverse_geocode(self.current_position)
location_str = location_info.get('address', {}).get('freeformAddress', 'Unknown location')
print(f"\n[PING] {zone.name}")
print(f"Distance: {distance:.1f}m")
print(f"Location: {location_str}")
```

```
print("-" * 50)
except Exception as e:
print(f"\n[PING] {zone.name} - Distance: {distance:.1f}m")

def _alert_inside_zone(self, zone: RestrictedZone):
"""Console output for inside zone alerts."""
print(f"\n🚨 ALERT: Inside Restricted Zone!")
print(f"Zone: {zone.name}")
print(f"Action Required: Please exit the restricted area immediately")
print("=" * 50)

def _alert_approaching_zone(self, zone: RestrictedZone):
"""Console output for approaching zone alerts."""
print(f"\n⚠️ WARNING: Approaching Restricted Zone!")
print(f"Zone: {zone.name}")
print(f"Action Required: Please be cautious of restricted area ahead")
print("=" * 50)

def start_watching(self):
"""Start monitoring location."""
if not self.watching:
self.watching = True
self.monitor_thread = Thread(target=self._location_monitor)
self.monitor_thread.daemon = True
self.monitor_thread.start()

def stop_watching(self):
"""Stop monitoring location."""
self.watching = False
if hasattr(self, 'monitor_thread'):
self.monitor_thread.join()

def _location_monitor(self):
"""Monitor location in a separate thread."""
while self.watching:
current_position = self.get_current_location()
if current_position:
self._check_zones(current_position)
time.sleep(0.1)

def test_geofencing():
"""Test function for the geofencing system."""
print("Starting Geofencing System Test...")
# Initialize the system
```

```
geofencing = GeofencingSystem(API_KEY)
# Test coordinates (New York City area)
test_zones = [
{
"center": Point(lat=40.7128, lng=-74.0060), # NYC
"radius": 100,
"name": "Times Square Zone"
},
{
"center": Point(lat=40.7527, lng=-73.9772), # Empire State
"radius": 200,
"name": "Empire State Zone"
}
]
# Add test zones
for zone in test_zones:
zone_id = geofencing.add_zone(
center=zone["center"],
radius=zone["radius"],
name=zone["name"]
)
print(f"Added zone: {zone['name']}")

# Test movement path
test_points = [
Point(lat=40.7128, lng=-74.0060), # Start at Times Square
Point(lat=40.7129, lng=-74.0061), # Move slightly
Point(lat=40.7130, lng=-74.0062), # Continue moving
Point(lat=40.7527, lng=-73.9772), # Move to Empire State
Point(lat=40.7528, lng=-73.9773), # Move slightly away
]

print("\nStarting movement simulation...")
geofencing.start_watching()
try:
for i, point in enumerate(test_points):
print(f"\nMoving to point {i+1}/{len(test_points)}")
geofencing.current_position = point
geofencing._check_zones(point)
time.sleep(3)
except KeyboardInterrupt:
print("\nTest interrupted by user")
finally:
geofencing.stop_watching()
print("\nTest completed")
```

```
if __name__ == "__main__":
test_geofencing()
```

For the bandwidth API multiple iterations of updates were used to be able to send the appropriate restrictions to the chosen devices, it was finalized with Shabodi bandwidth API. The iterations were done with the input of Claude AI with the initial prompt being:

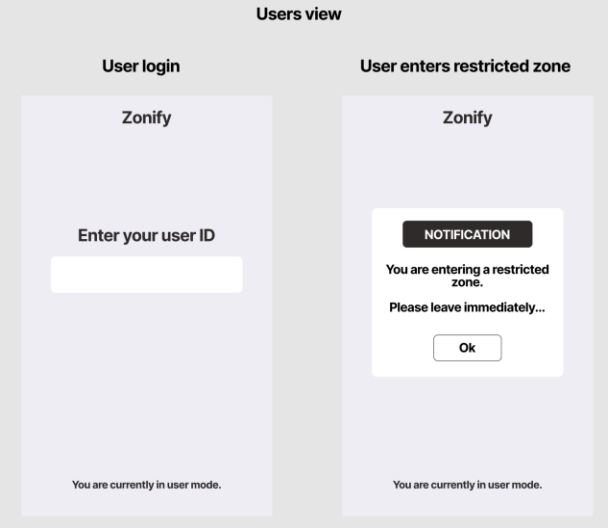
using a bandwidth API with room for token checking, can you make a code in python that when, receiving a sign from a different function that will limit all bandwidth other than the current application?

And given the code:

```
import requests

client_id = "380dfca1-6539-4890-9595-21a57c8f907d"

client_secret = "enMKQcRauITqqdsDGsbNDUN_JlNgLYQkkdPhe5IF8Ws"

aep_host = "192.168.3.18"  # Shabodi server IP



# Function to request the access token

def get_token(client_id, client_secret):

    url = f"http://{aep_host}:31002/security/v1/token"  # Adjusted port and endpoint

    headers = {

        "Content-Type": "application/json"

    }

    data = {

        "client_id": client_id,

        "client_secret": client_secret

    }
```

```
    # Making the POST request to obtain the token
    response = requests.post(url, headers=headers, json=data, verify=False)
    if response.status_code == 200:
        token = response.json().get("access_token")
        print("Access Token:", token)
        return token
    else:
        print("Failed to retrieve token:", response.status_code, response.json())
        return None


# Run the function to get the token
if __name__ == "__main__":
    token = get_token(client_id, client_secret)
```

Finally, for the user interface it was initially created on Figma to decide on a design and Claude AI was used to bring it as a Python program. With the initial designs being:



The prompts given to Claude AI to create the application code are as follows:

6.1 Bill of Materials

Beyond the necessary hardware, such as laptops and internet connection, no other purchase was necessary. However, 1 month of Claude AI was bought to expediate the progress of the team in program development. If needed future development may require additional software or specialized hardware.

Item #	Name of service	Cost
1	Claude 1-month Pro Plan	$20

6.2 Testing & Validation

Location APi Pinging Users

Objective: Ensure the application notifies when a user enters a restricted zone

Method: Utilized AI on Shabodi sandbox to simulate user enter a restricted zone.

Result: The Ai confirmed that the code successfully pings the user at the right increments the closer it gets to the restricted zone and as the user enters the restricted zone.

Functional User Interface (UI)

Objective: Ensures the UI is functional for user and admin

Method: any error message the code produced, the AI-driven scripts would fix and retest the interface by simulating user and admin interactions.

Result: The Ai verifies that user and admin could type in codes and access their respected feature without issue

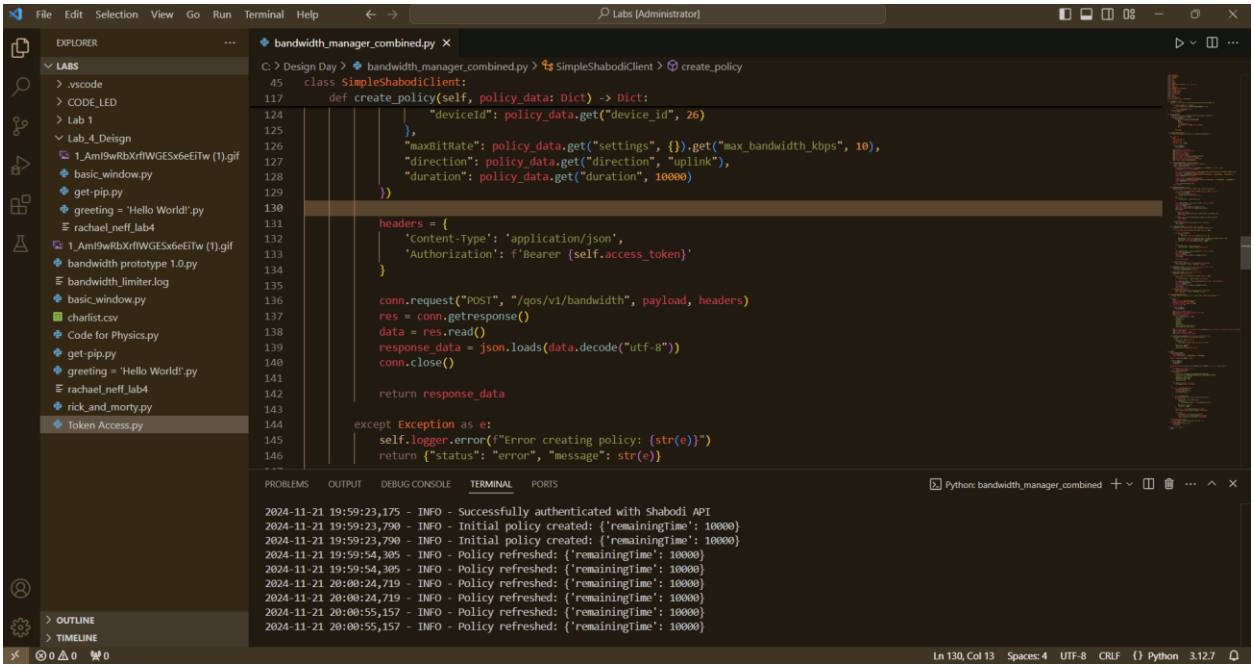
```
/Users/williamgillespie/PycharmProjects/userIntTest/.venv/bin/python /Users/williamgillespie/PycharmProjects/userIntTest/errorTest.py
Traceback (most recent call last):
  File "/Users/williamgillespie/PycharmProjects/userIntTest/errorTest.py", line 1, in <module>
    import customtkinter as ctk
ModuleNotFoundError: No module named 'customtkinter'

Process finished with exit code 1
```

Demonstrating Shabodi Bandwidth Functionality

Objective: Ensure the application bandwidth usage is effective

Method: Conducted by using the Shabodi sandbox to simulate devices connected to the network.

Results: The device would respond with the new bandwidth limitation and length of time it would be limited for.



7 Conclusions and Recommendations for Future Work

In particular, the most important lessons were dedicate more time towards the prototyping phase of the project, receive proper direction from the client before dedicating as much time towards the design phases.

A couple concepts such as multi-floor capability was abandoned due to time constraints and API limitations. If more time was allocated, it would have been put towards a single functional application instead of multiple separate functions.

8 Bibliography

Claude. https://claude.ai/new

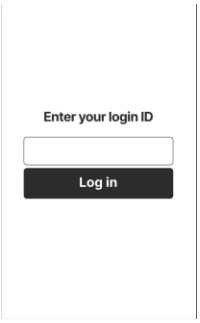
Location API Prompts with dates

10-31-2024
I need you to create a code that uses a location API. This code will be used to create restricted zones, when a restricted zone is created the code needs to be able to alert a user not only when they are approaching a restricted zone but also when they are in one. There should be two different sounds and vibrations for those notifications. We should also be able to change and customize a restricted zone
complete this code in python.
can you add in a pinging section, where it will ping your location the closer you are to the restricted zone and it will ping it less the further you are away from the restricted zone
write it using this API key **47C4ECCCB1EF134EB784BDB1660FD90B**
It is for trimble maps and this is a publically available API
can you fill in some random info in the blank parts so that I can test the code
11-3-2024
can you take the code you have already given me but take out the pygame and the notification portion because that has been causing me issues\
Now can you add a notification system that will notify a person when they are approaching and when they are in a restricted zone, can you try not to use pygame because i couldn't get that to work on my computer
in line 53 i am getting an invalid syntax error
it says no module names modify two
11-13-2024
can you replace all of the places with tremble mentioned with a place to input our own api information
the api i am going to use is a location api so I do not have a different api maps key
11-20-2024

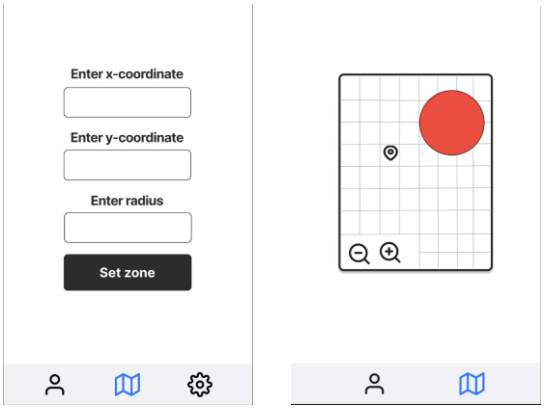
Can you change this code so that it is set up to utilize googles geolocation api:
where do I incert googles api key
can you highlight where that is in my code
i dont see that in my code can you add it
idx says that no googlemaps module found
how can i connect my idx code with my google cloud account so i can use the api
will this allow me to use the google api in my idx code
i am getting this error API Error: REQUEST_DENIED (This API project is not authorized to use this API.)
i am not able to select my idx project though i can not find it
can you ensure that the test section of my code does no need the use of a speed api
Yes

Application GUI prompts with dates:

11-15-2024
Can you make me an app in python using pQyT5 that looks like the picture, where the writing for the login is black and the black "Log in" button is an interactive button



now can you change the code so that if the user enters 1111 and presses log in, it switches to another screen that looks like the first picture (where the writing is in black or visible). If the user enters anything other than 1111 and presses Log in, it switches to a screen that looks like the second screen where there is an interactive map.

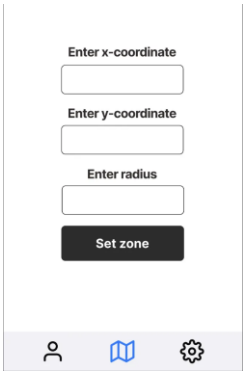


can you make the + and - buttons zoom in and zoom out

in the interactive map screen, can you make an option that allows the user to return to the log in screen

11-19-2024
can you modify this code so that there is a 'back' option on every screen (except the initial log in screen

Now can you add a settings option in this screen. In the settings screen, can you make an option to toggle a button called 'bandwidth' on or off



there is an error warning: Traceback (most recent call last): File "/Users/williamgillespie/PycharmProjects/userIntTest/updatedUI.py", line 2, in <module> from PyQt5.QtWidgets import (QApplication, QMainWindow, QWidget, QVBoxLayout, ImportError: cannot import name 'QSwitch' from 'PyQt5.QtWidgets' (/Users/williamgillespie/PycharmProjects/userIntTest/.venv/lib/python3.9/site-packages/PyQt5/QtWidgets.abi3.so)

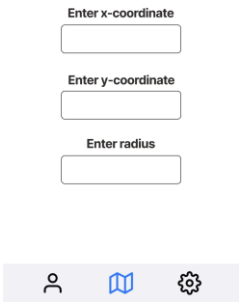
can you change this code so that the settings button is only available on the coordinates screen? not the map screen

now can you add another map screen with one big red circle and multiple small white circles nearby after the user presses the 'set zone' button

Can you modify this code so that the small circle in the map is white and the title of the GUI is 'Zonify'.

11-22-2024

Can you change the colour of the background of this app to one similar to the image provided? (the colour codes are FFFFFF, F2F2F7, and 2F2B2B)



Now just make the writing in the text boxes black

There is a problem with the map screen as there is a white screen blocking the circles in the back and the + and - sign should be black as well

Now can you add a zoom in and zoom out function in the map

The program works, but I get these messages:
/Users/williamgillespie/PycharmProjects/userIntTest/updatedUI.py:97:
DeprecationWarning: an integer is required (got type float). Implicit conversion to integers using **int** is deprecated, and may be removed in a future version of Python.
painter.drawEllipse(self.x - self.radius, self.y - self.radius,
/Users/williamgillespie/PycharmProjects/userIntTest/updatedUI.py:104:
DeprecationWarning: an integer is required (got type float). Implicit conversion to integers using **int** is deprecated, and may be removed in a future version of Python.
painter.drawEllipse(circle_x - small_radius, circle_y - small_radius, I also need you to change the colour of the text on the settings screen to black

In the settings screen, I want there to be the word 'Bandwidth' in black next to the toggle button. Can you add to the code

Bandwidth prompts with dates:

11/13/24

using a bandwidth API with room for token checking, can you make a code in python that when, receiving a sign from a different function that will limit all bandwidth other than the current application?

11/20/24

Import the following code into the bandwidth code:

```
       import requests

client_id = "380dfca1-6539-4890-9595-21a57c8f907d"

client_secret = "enMKQcRauITqqdsDGsbNDUN_JlNgLYQkkdPhe5IF8Ws"

aep_host = "192.168.3.18"  # Shabodi server IP



# Function to request the access token

def get_token(client_id, client_secret):

    url = f"http://{aep_host}:31002/security/v1/token"  # Adjusted port and endpoint

    headers = {

        "Content-Type": "application/json"

    }

    data = {

        "client_id": client_id,

        "client_secret": client_secret

    }

          # Making the POST request to obtain the token

    response = requests.post(url, headers=headers, json=data, verify=False)

    if response.status_code == 200:

        token = response.json().get("access_token")

        print("Access Token:", token)

        return token
```

```
    else:
        print("Failed to retrieve token:", response.status_code, response.json())
        return None

# Run the function to get the token
if __name__ == "__main__":
    token = get_token(client_id, client_secret)
```

9 APPENDICES

APPENDIX I: Design Files

Deliverables - Can be found on MakerRepo.

Link to MakerRepo Project Page: GNG1103-P16-Zonify | MakerRepo