Project Deliverable H: Prototype Ⅲ & Customer Feedback

Abby, Nava, Neyssa, Aadi, Gill

Abstract

*This deliverable focuses on the development and iterative testing of Prototype III and the collection of customer feedback. It includes detailed documentation for the development, testing, and analysis of Prototype III, as well as feedback from potential users or clients to refine and improve the design. Updates to target specifications, detailed design, and the bill of materials are addressed to ensure alignment with project goals. A comprehensive prototype and test results are provided to validate the design's functionality. This deliverable also serves as a foundational step towards preparing a user manual and achieving the final project deliverables.*

Table of Contents

[**Introduction 3**](#_uo1v4mstxso5)

[**Prototype Development 3**](#_u40nvl7t94qd)

[Documentation for Prototyping 3](#_1o7n56acrug0)

[**Critical Components Analysis 11**](#_w7dye1cnrd6p)

[**Gathering Feedback 12**](#_53z2yd4k3ckb)

[**Updates 13**](#_43eikup6yldy)

[Target Specifications 13](#_t3zy3gk1fnxi)

[**Appendix 16**](#_um96s844tikx)

# Introduction

In the last deliverables, our team discussed the developments of prototype one and two for our SIM swap detection application. Those initial prototypes focused on building the framework of the application and GUI refinement. This deliverable aims to carefully document the development of our third prototype where we finally integrate Shabodi’s API’s, add a few more elements to the GUI, test all of the core subsystem functionalities to ensure the product is aligning with target specifications and obtain customer feedback on the prototype.

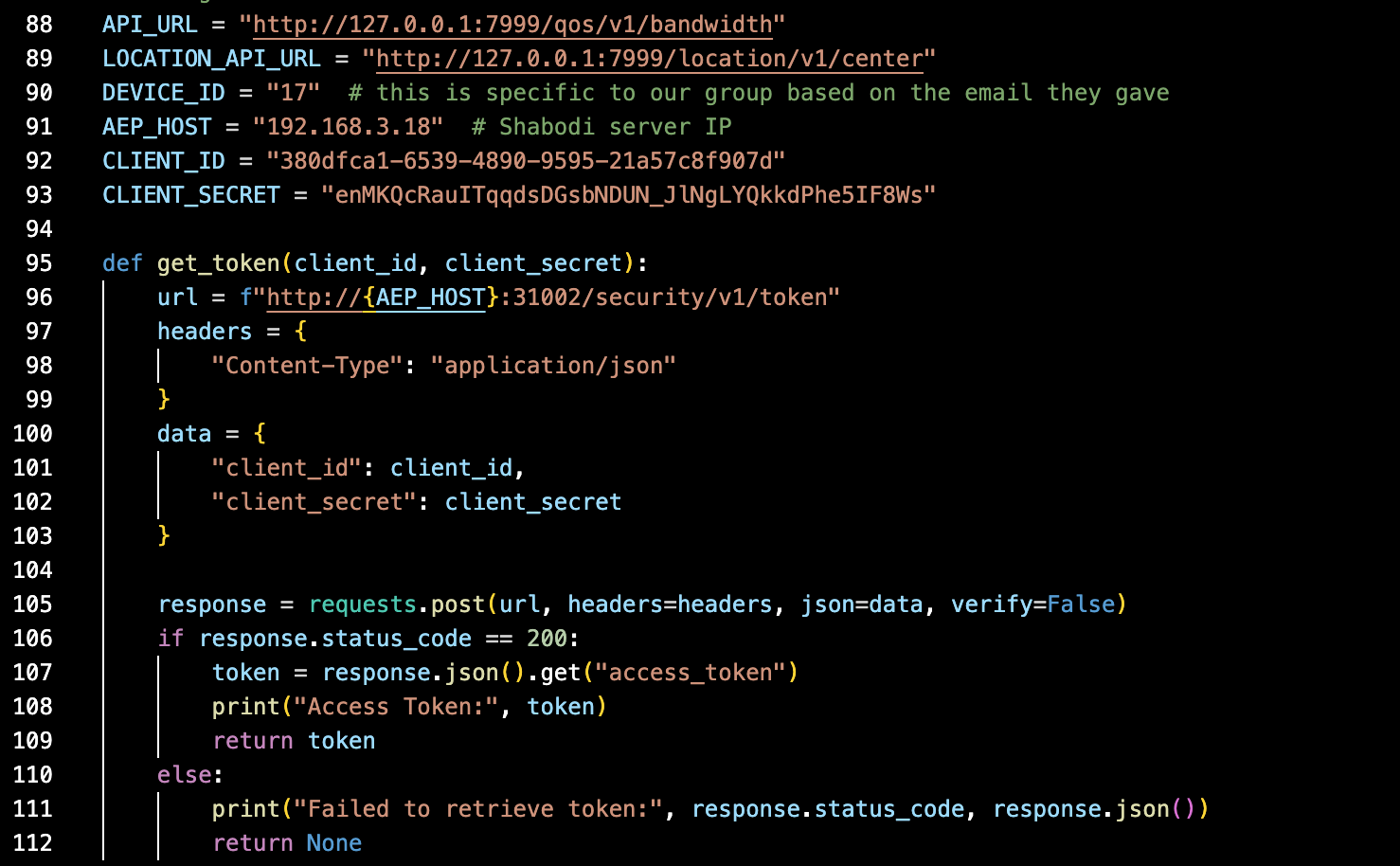
# Prototype Development

## Documentation for Prototyping

This week, the main priority for this prototype was to finish off the minimum viable product (MVP) work and ensure that all of Shabodi’s APIs were fully and correctly integrated into the system. The functionality and operability of the system was tested as per the Prototype Test Plan Outline detailed in the next section. As per the documentation, specific and notable changes will be detailed below.

1. Shabodi API Token Generation Code

One of the most important tasks after some of us were able to access Shabodi’s Sandbox was to make sure that the proper code to generate an API token was written. Although a fully detailed copy of the code from these files is in the **Appendix**, a snippet of the specific token generation code is written below:



Our group was able to do this thanks to cross-referencing the code provided by Group 6 on the Brightspace. This code allows the program to interact directly with the Shabodi server and successfully access the APIs.

1. Integration of Shabodi’s APIs and functionality testing

Our group was able to successfully integrate the Location, Bandwidth, SIM Swap, and Latency APIs in the code. They appear in both our interface file and our logic file, as shown in the Appendix. There are too many instances of their integration into the program to take a screenshot, but they are implemented comparably to the token generation code using similar logic.

1. Updating in-program documentation

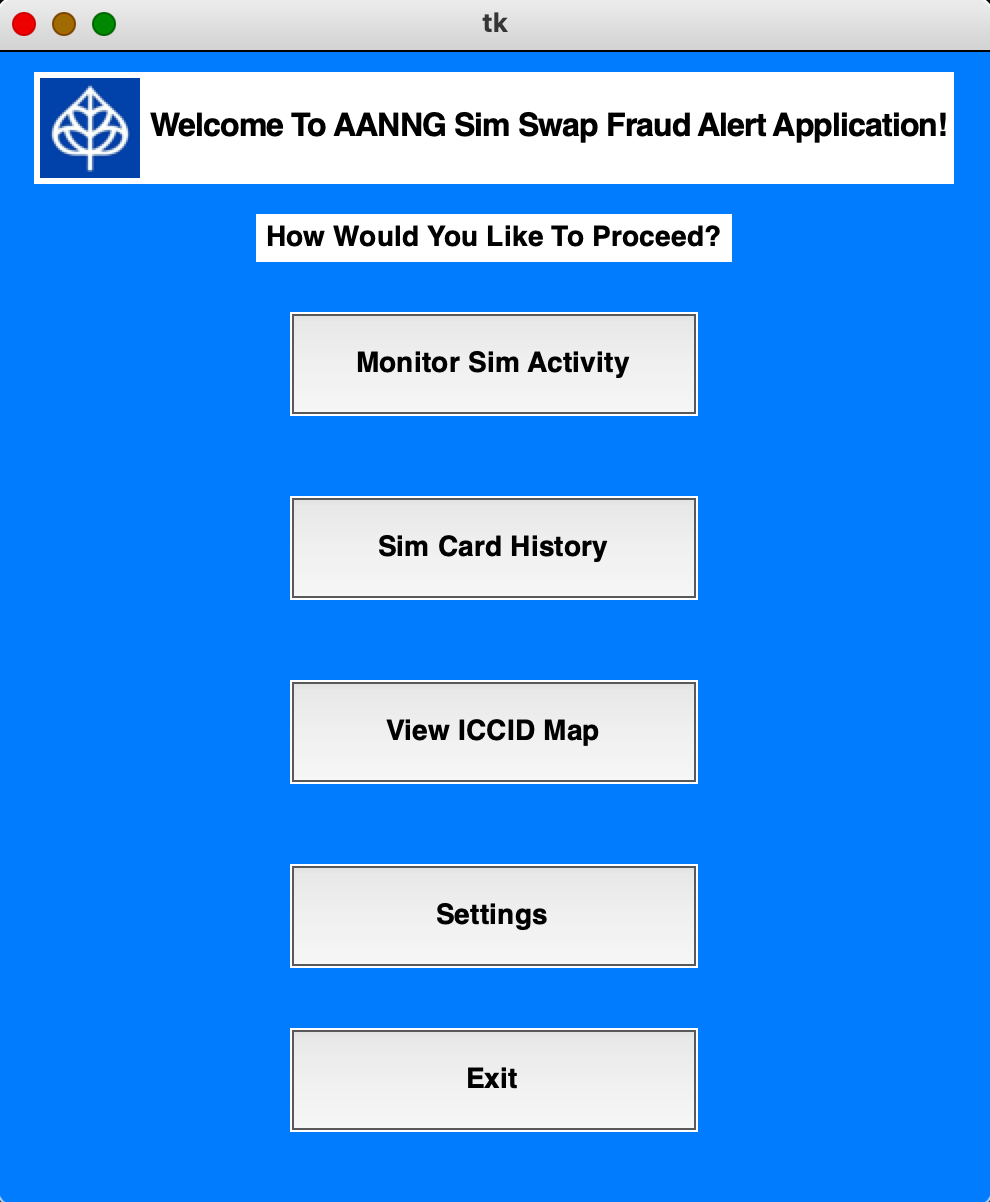
Comments were added to the code for both specific blocks and lines to ensure readability. This was also done with the intention of preparing the code to be able to be understood by those without a programming background for the User Manual.

1. Changes to GUI operation

After the successful integration of Shabodi’s APIs into the code, one of the major updates was related to the GUI. The GUI was updated to reflect these changes and allow for ease of testing with respect to these APIs. This is detailed further in the Prototype Test Plan Outline detailed below.

The GUI is a high-fidelity comprehensive aspect of the system which now represents the MVP version of the application. Screenshots of relevant changes to the GUI are depicted below; full code is available in the Appendix:

Figure 1: Overall GUI Access Image



In the overall GUI which opens instantaneously after running the program, the main change at this interface point was the addition of the exit button which terminates the system immediately.

Figure 2: SIM Card Monitoring Live Feed

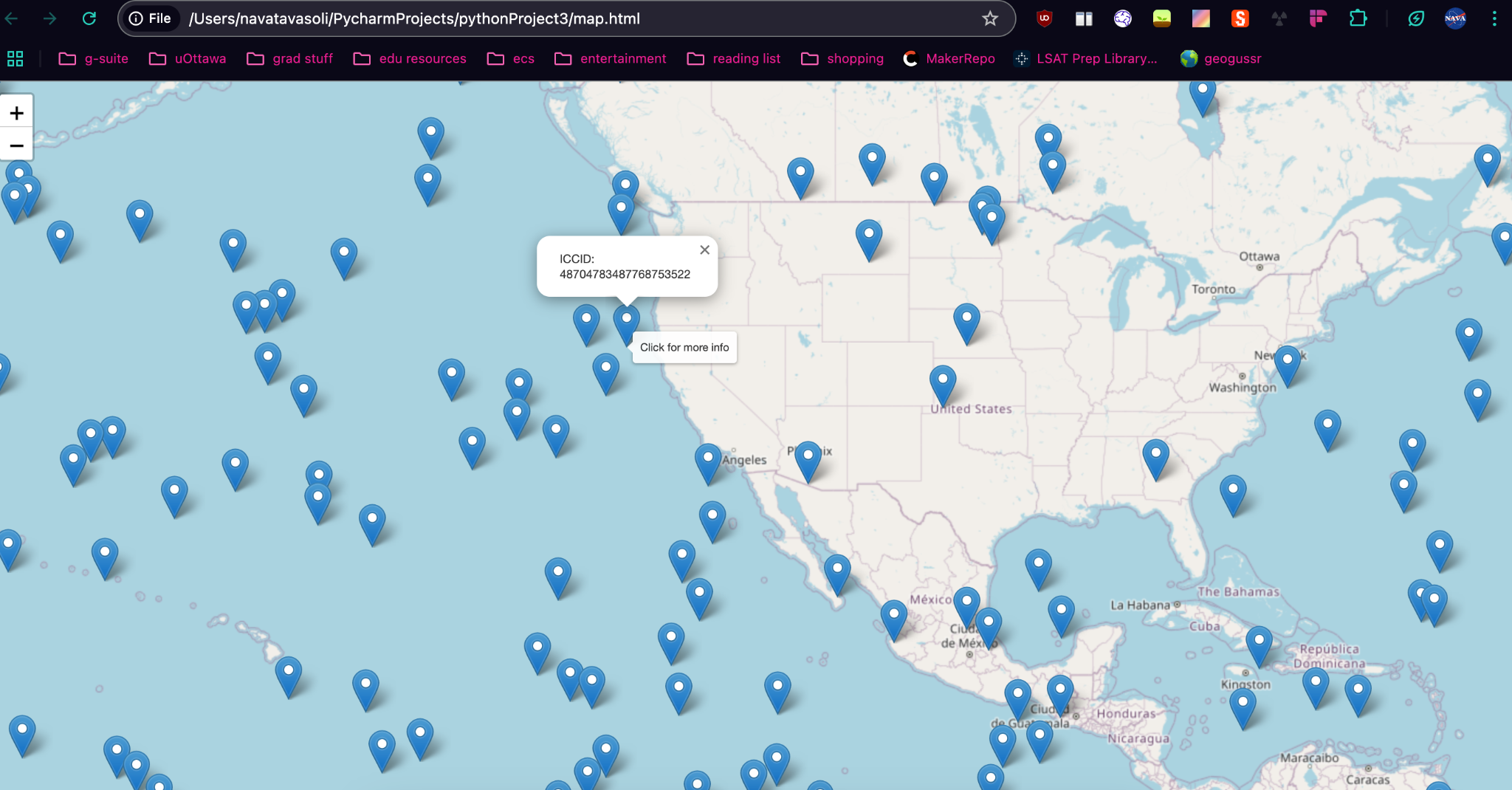
When a user presses the first button, a tab will open providing real time data of the current SIMs under the enterprise’s network from the relevant JSON file.

Figure 3: SIM Alert History Tab



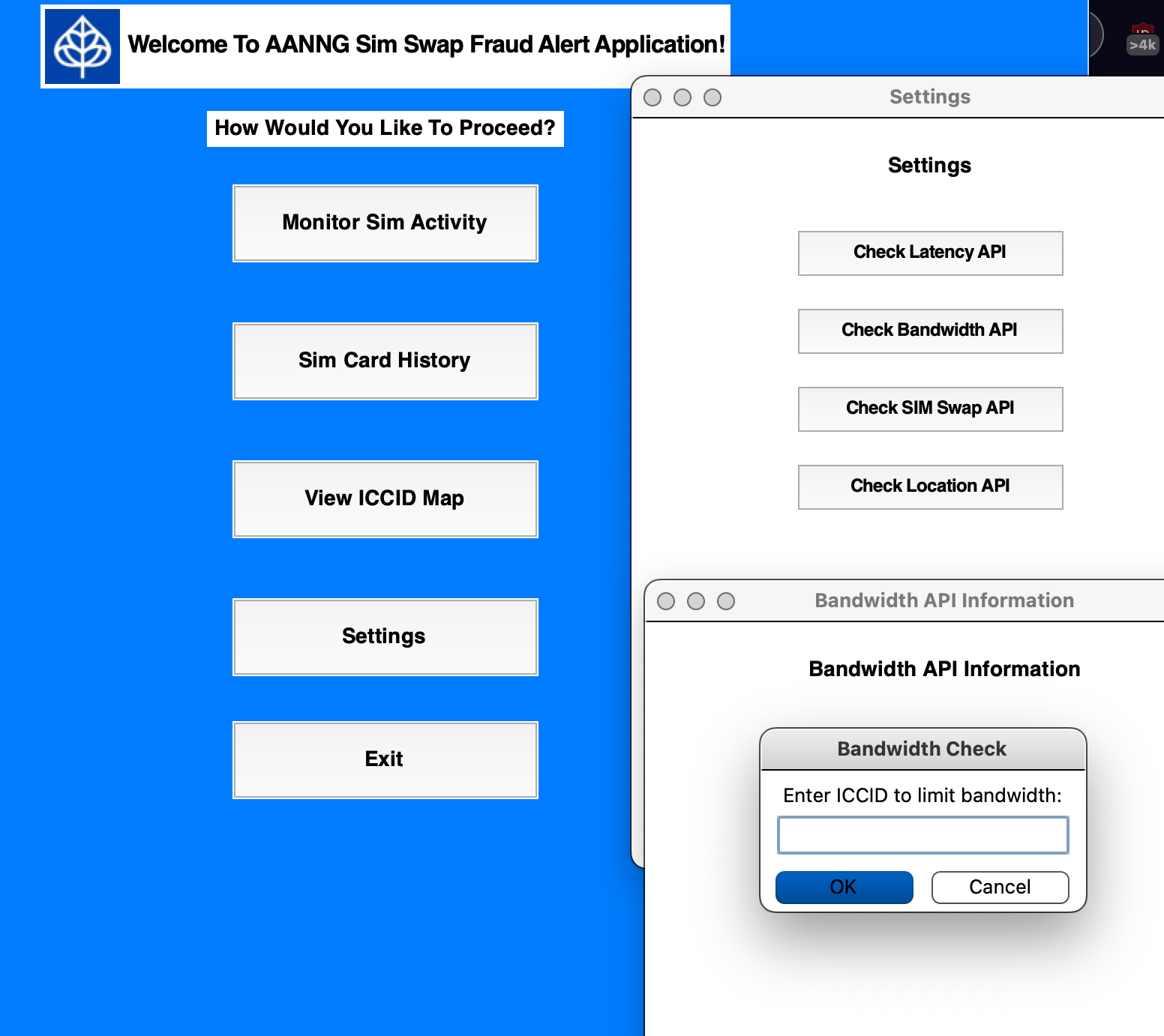
The SIM Card History button provides a list of data from previous alerts, when SIM swap/duplication was noticed in the past. This tab provides the location, date, and time of a SIM fraud event that was recorded.

Figure 4: ICCID Live Map Button



Pressing the view ICCID live map button is the same as it was in prototype 2. The user will be taken to a live map in their default browser where they can see all ICCIDs under the enterprise’s 5G network. They can click on each icon to get the exact ICCID information. Over 3 times the expected number of devices were used in the mock JSON file, as seen in the image, for testing purposes.

Figure 5: Settings Button (Bandwidth API example)



In the Settings menu, the user can access immediate information on the status of each of the APIs by entering in a specific ICCID. In this figure, the Bandwidth API button is shown as an example.

Prototype Test Plan Outline

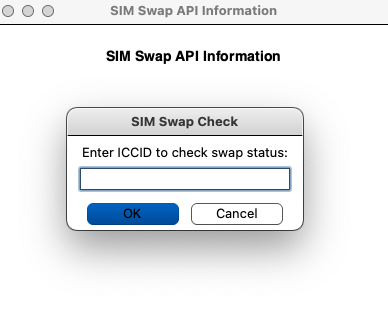
As we approach the completion of our MVP, our group has adjusted our prototype test plan outline to be more suited to the specific items we would like to test in this final prototype. Our group members have gained access to Shabodi’s Sandbox and VPN after heavy trial and error. Our planned tests are specifically designed to ensure a seamless integration of these APIs into the preexisting code from Prototype II.

Each of the tests have been divided into specific sections for clarity and thoroughness of testing details. Nearly all of the comprehensive and focused prototype tests in this stage of development are high fidelity to mimic the functionality of the MVP.

1. **Core SIM Swap/Duplication Detection Algorithm Testing**

| **Subsystem Target/Technical Specifications** | **Test Goals/Updates from Prototype II Testing** | **Objectives and Notes** | **Estimated Duration** |
| --- | --- | --- | --- |
| HiFi comprehensive | * Test the accuracy of the core detection algorithm for both swaps and duplications | * Shabodi APIs used:   + SIM Swap API | From completion of functional code to 24/11/2024 with iterative testing on each day. |

We will be testing the core detection algorithm as a high-fidelity focused prototype. We will be using the SIM Swap API from Shabodi’s NetAware Sandbox for this purpose. For testing purposes, and as a component of the MVP, we have added the following button in our ‘Settings’ section of the GUI in order to interactively display information to a user on the successful acceptance/declining of their SIM Swap which will confirm that the detection mechanism works properly:



A pop-up notification will inform the user of the outcome of their swap detection for their specified ICCID. This is an integral part of the testing process, since it will allow us to make sure that our core detection algorithm functions properly and is able to detect both duplication and swap.

Our group emulated a SIM duplication in real time using a Network JSON file, and a screenshot of the duplication (for testing purposes) is seen below:



It is also important to note that our team used nearly 7000 lines in the JSON file; this is above the target specification identified by Shabodi, and this was done on purpose to ensure that the compilation time of the program is sufficiently equipped to handle enterprises of different sizes as per the desire of Shabodi. We will continue to iteratively test the core detection algorithm subsystem as a comprehensive and high fidelity prototype using the testing mechanisms described.

1. **Bandwidth Choking Mechanism / Administrative Alert Subsystem Testing**

The Bandwidth Choking mechanism is a new subsystem introduced for implementation in this specific prototype. It aligns its function with the Administrative Alert subsystem. This has been implemented as per Shabodi’s suggestions to have a more high-security approach to the application.

| **Subsystem Target/Technical Specifications** | **Test Goals/Updates from Prototype II Testing** | **Objectives and Notes** | **Estimated Duration** |
| --- | --- | --- | --- |
| HiFi focused | * Test the functionality of the pop-up notification   + Ensure the notification appears after every detected duplication/swap   + Ensure it appears within **2 minutes** after fraud is detected   + **The delay between detection and notification must be displayed to the user** * When a user is prompted, ensure that they have the option to choke the bandwidth of the fraudulent device to prevent further damage to the enterprise | * Shabodi APIs used:   + Bandwidth API   + SIM Swap API   + Latency API | From successful testing of SIM Swap core detection algorithm to 22/11/2024. |

1. **Location Detection Subsystem Testing**

| **Subsystem Target/Technical Specifications** | **Test Goals/Updates from Prototype II Testing** | **Objectives and Notes** | **Estimated Duration** |
| --- | --- | --- | --- |
| HiFi focused | * Test the functionality and operability of the ICCID mapping   + Use **3 times the expected number of ICCIDs** * Ensure that compiling time is within reason | * Shabodi APIs used:   + Location API | From successful testing of SIM Swap core detection algorithm to 24/11/2024. |

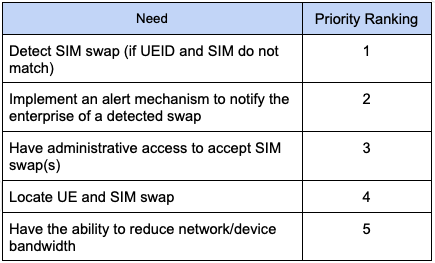
1. **Latency API Integration Testing**

Given the issues that our team and many others faced with the integration of Shabodi’s APIs into our program, we wanted to take special care to test the focused prototype of our Latency API in the system.

| **Subsystem Target/Technical Specifications** | **Test Goals/Updates from Prototype II Testing** | **Objectives and Notes** | **Estimated Duration** |
| --- | --- | --- | --- |
| HiFi focused | * Ensure that, in accordance with the Shabodi training material guidelines, the Latency API will **set and retrieve** latency session information for a specified ICCID. | * Shabodi APIs used:   + Latency API   + Location API | From successful testing of SIM Swap core detection algorithm to 24/11/2024. |

This testing plan has been updated most notably since Deliverable G (Prototype II) since it carries a less urgent priority ranking than the other APIs in terms of its importance for the overall functionality of the minimum viable product.

The entirety of the prototype testing plan and all of the updates made considered the Needs Prioritization of Shabodi done in one of the first Deliverables after Client Meeting #1:



# Critical Components Analysis

| **Critical Components / Systems** | **Analysis** |
| --- | --- |
| Authentication Mechanism | * **Strength**: provides security and privacy to the user’s SIM card number * **Potential Vulnerabilities:** token theft or interception of the data * **Improvement**: create an implementation of token refresh mechanism that will securely store tokens |
| SIM Swap Detection API Integration | * **Strength**: Direct communication with network for real-time detecting * **Potential Vulnerabilities:** we are dependent on a third-party * **Improvement**: implementation of a “fallback” mechanism to check the health of the API |
| Phone Number Validation | * **Strength**: prevents invalid inputs of numbers * **Potential Vulnerabilities:** could be a possible way to bypass this mechanism * **Improvement**: consider international number formats to increase unique patterns |
| Error Handling | * **Strength**: helps debug and increase our security within the code * **Potential Vulnerabilities:** possible information leakage or errors when handling the mechanisms of the program * **Improvement**: create a structured logging to ensure the sensitive data is not logged and can’t be viewed by the public |
| User Interface & SIM Swap Alerts | * **Strength**: informs user of security threats and allows usability for the client/user * **Potential Vulnerabilities:** if we alert the user too often, it can cause alert fatigue. Also potential crashes of the application if not maintained. * **Improvement**: make sure the alerts are only active when there is a real threat and that the application is running smoothly at estimated seconds |
| Location API integration | * **Strength**: ability to track location of SIM Swap activity * **Potential Vulnerabilities**: sensitive personal information can be exploited in case of a breach * **Improvements**: Operate according to the Principle of Least Privilege, where the application has access to only what it needs form users |
| Bandwidth API integration | * **Strengths**: assessing real-time performance, its capacity to evaluate the system. Ability to customize and adjust the bandwidth of a suspicious user * **Potential Vulnerabilities**: Possible latency between the time a SIM swap is detected and when the bandwidth restrictions are applied. * **Improvement**: Ensure that fraud detection and bandwidth management systems work in as close to real-time to reduce delays |
| Latency API Integration | * **Strengths**: Determine lag time between alert notification and event detection. * **Potential Vulnerabilities**: calculations could be wrong or misinterpreted * **Improvement**: ensure the integration has no bugs and that the API is maintained within the code efficiently |

# Gathering Feedback

| **Client Name** | **Potential User Type**  **(Perspective)** | **Likes/Comments** | **Additional Feedback** |
| --- | --- | --- | --- |
| Rohith Vidyasakar | Potential user | * The relative simplicity of the application makes it easy to navigate between menus * The ICCID map is a nice addition quickly giving the user an update on SIM statuses * Some UI elements are slightly un-optimised, however taking into account the importance of functionality over UI, the final prototype is still well realized. | * From a UI point of view : the text is a bit small and wordy with somewhat inconsistent margins and aspect ratio which contribute to a slightly dated look. * Also, the addition of numbered IDs (one number for each SIM ID) might increase the ease of recognition and differentiation as there could be many in one location. |
| Ian Trites | Potential user | * Simple GUI design so as not to overload with information and options (only 4 buttons) * Appreciates the smooth color choices * Elimination of One Time Passwords (OTP) and Two Factor Authentication (TFA) as well as data encryption is appreciated * However potential implementation of efficient and reliable biometric “sign in” might prove to be a slight challenge | * Some kind of help should be accessible in the main page of application → link towards User Manual or explanation on how to use product |
| Shoko Trites | Potential user | * Simple menu navigation with only one click to link multiple pages is appreciated * Location tracking capability is an interesting way to increase SIM security | * The ICCID map is nice however it could include color coded pindrops with (coded depending on the last digit of the ID) so as to improve differentiation. |

Feedback from varying sources has been a vital part of our project and served to significantly improve its quality.

Overall, the feedback returned was positive, reflecting the ease of use and functional properties of the application. Our project is now in the final stages of integrating APIs and quickly moving towards a minimum viable product. Some elements relating to the UI were flagged and noted, however given the importance of functionality over UI, our design still meets requirements. Some clarifying changes could be made to the ICCID map or main GUI page. That being said, within feasible levels (given the time left) we will look into these last elements as we finalize our project and prepare for Design Day.

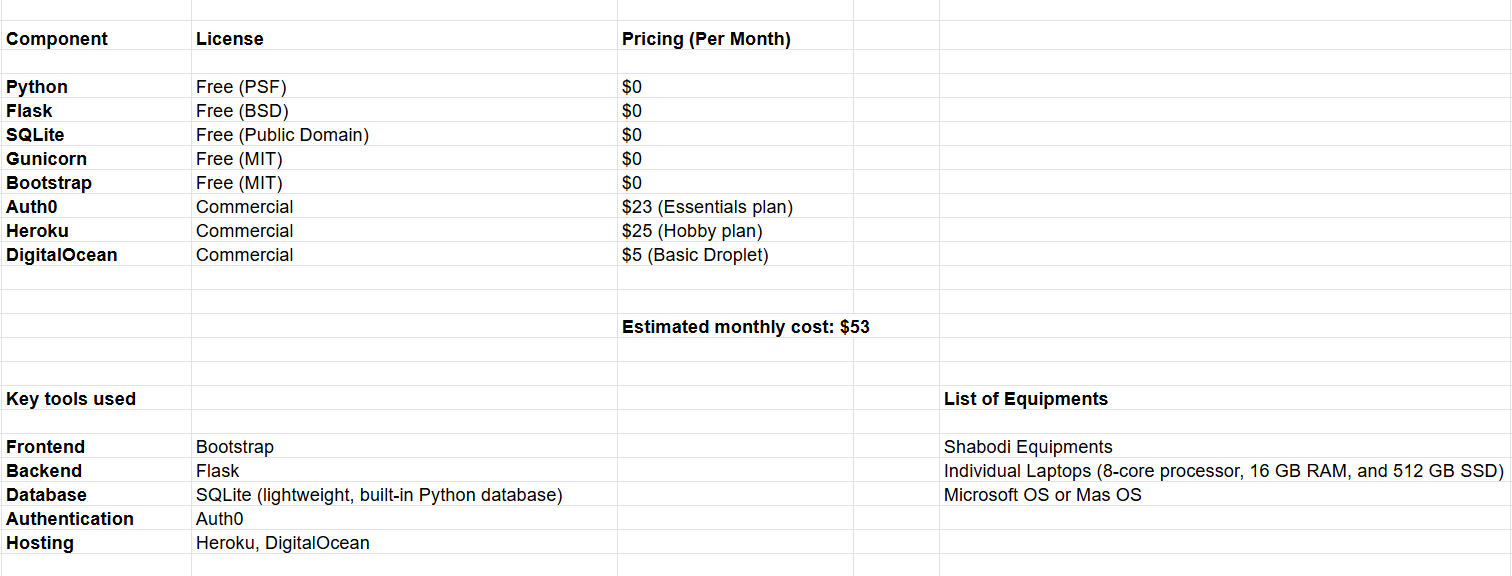
# Updates

## Target Specifications

The target specifications have been updated to reflect the enhancements made in functionality and performance. Shabodi's APIs, including SIM Swap, Bandwidth, Latency, and Location, have been fully integrated into the system, significantly improving detection accuracy through repetitive testing. High-fidelity testing was conducted using mock data with over 7,000 JSON entries, ensuring the system can handle large scale operations above the original target. Additionally, the latency between fraud detection and notification has been optimized to under two minutes. The user interface now features interactive alerts for real-time SIM monitoring and added options for bandwidth choking and latency checks, providing a comprehensive and user-friendly experience.

Bill of Materials

The updated bill of materials includes both hardware and software components essential for the project's success. The system requires a device with a minimum configuration of an 8-core processor, 16 GB RAM, and 512 GB SSD, along with networking equipment to gain access to Shabodi's VPN and sandbox. Software components include Shabodi's APIs, Python 3.9 with libraries such as tkinter and requests, and tools like Folium for ICCID mapping and the Shabodi SDK for API interactions. The testing framework uses JSON mock files and error-handling mechanisms to ensure the system’s reliability.



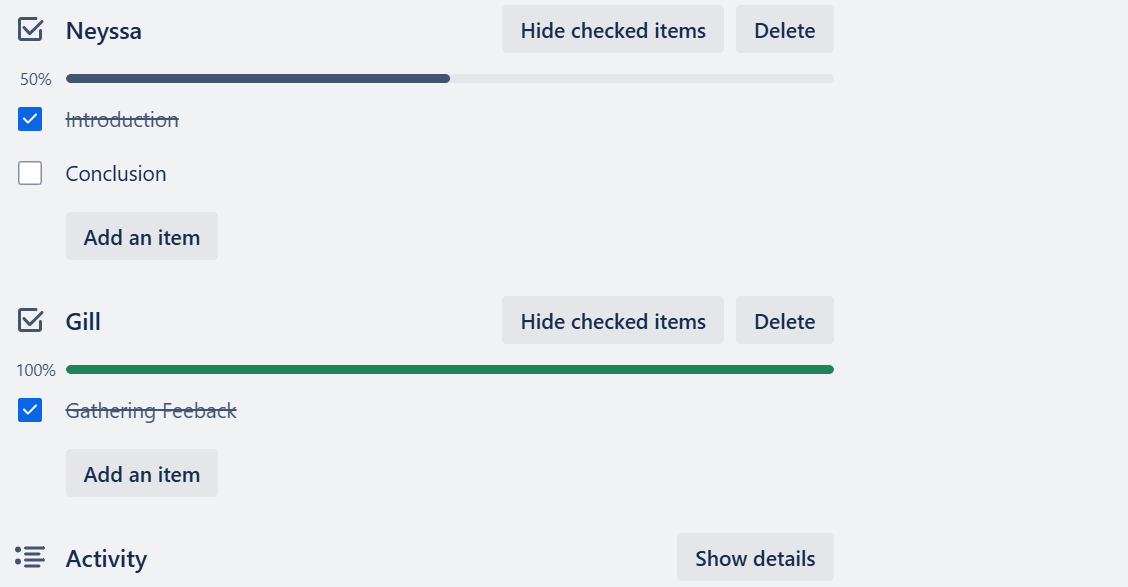
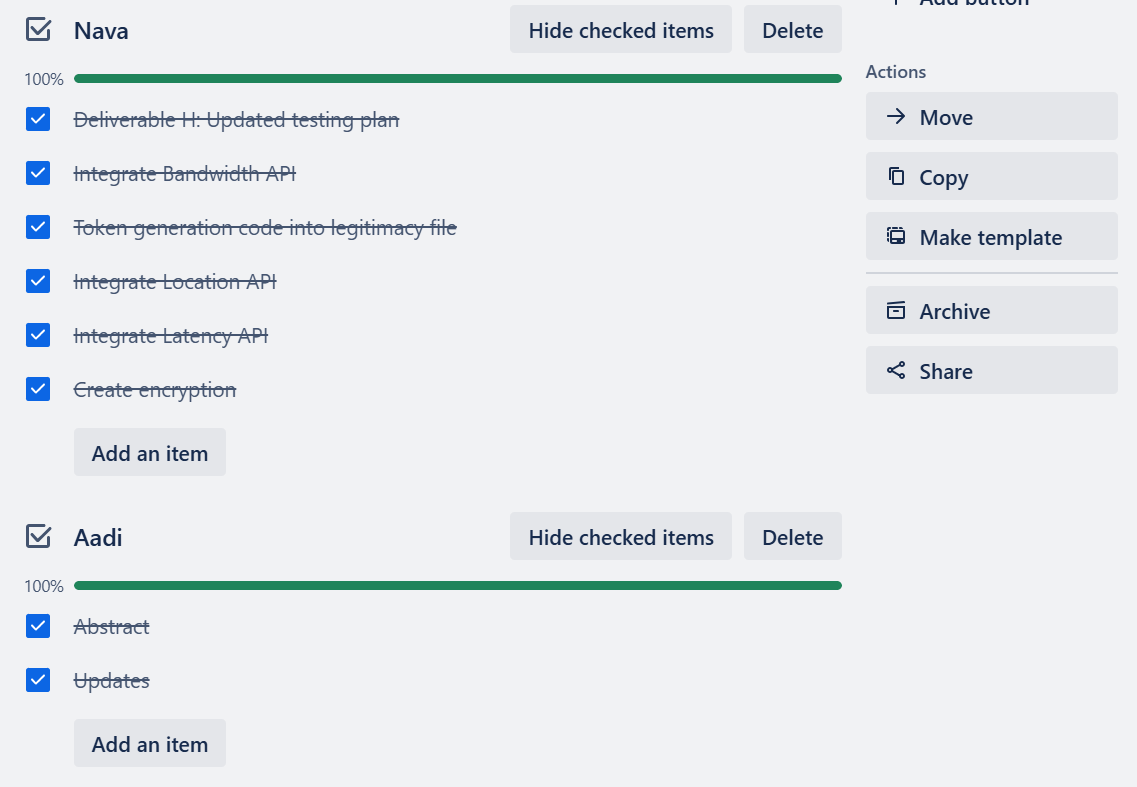
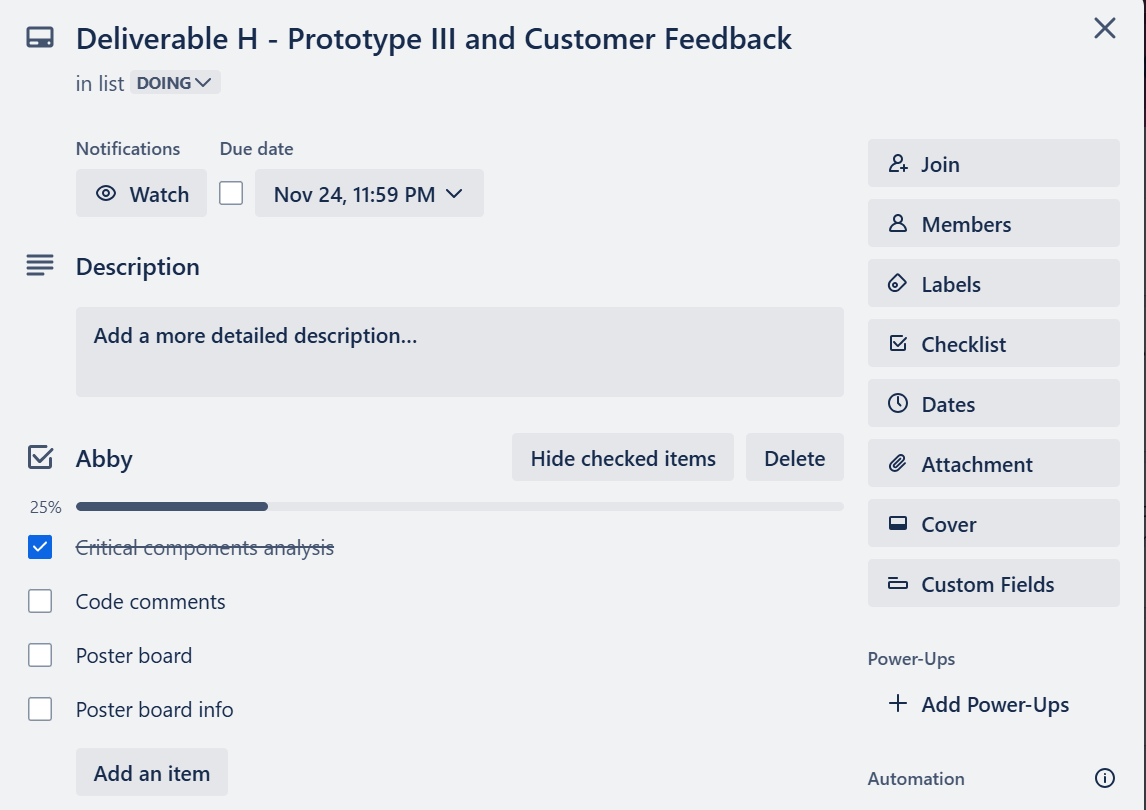
As an additional note for documentation purposes, we have created a table of all Python libraries that were used in the application. This is also reflected as part of the SBOM:

| **Library** | **Use Case File** |
| --- | --- |
| json | SIM\_swap\_legitimacy.py AND AANNG\_user\_interface.py |
| numpy/np | SIM\_swap\_legitimacy.py |
| time | SIM\_swap\_legitimacy.py |
| requests | SIM\_swap\_legitimacy.py |
| datetime | SIM\_swap\_legitimacy.py |
| cryptography and Fernet | SIM\_swap\_legitimacy.py |
| tkinter | AANNG\_user\_interface.py |
| messagebox and simpledialog | From tkinter, AANNG\_user\_interface.py |
| PIL | AANNG\_user\_interface.py |
| Image and ImageTK | From PIL, AANNG\_user\_interface.py |
| folium | AANNG\_user\_interface.py |
| os | AANNG\_user\_interface.py |
| webbrowser | AANNG\_user\_interface.py |
| requests | AANNG\_user\_interface.py |

Conclusion

In summary, this deliverable covered the development and documentation of our third prototype development. Shabodi’s APIs were successfully integrated into the code. Testing was initiated on all of our subsystems, including the detection algorithm, bandwidth choking mechanism and location detection subsystem. We were able to implement a prototype test plan and conduct product testing accordingly. A critical components analysis was also performed in this deliverable to evaluate the key components contributing to the products overall functionality. We ensured that we received feedback from a focus group of potential users to determine operability and functionality. Target specifications were also updated to reflect enhancements made in the performance and our Bill of Materials was updated as well. With the minimum viable product being completed, our team is currently focused on preparations for Design Day where we will display our final product to potential users and clients.

Trello Board (Task Distribution)



# Appendix

Direct code from file AANNG\_user\_interface.py:

| import tkinter as tk  from tkinter import messagebox, simpledialog  from PIL import Image, ImageTk  import folium  import json  import os  import webbrowser  import requests  # window configuration and greeting label  window = tk.Tk()  window.configure(bg="#017cfe") # shabodi color background  window.geometry("500x500")  # creating the Shabodi greeting image  greeting\_image\_path = 'C:/Users/Neyssa/Downloads/shabodi\_icon.jpg'  greeting\_image = Image.open(greeting\_image\_path)  resized\_image = greeting\_image.resize((50, 50))  photo = ImageTk.PhotoImage(resized\_image)  # greeting labels  greeting = tk.Label(  text=" Welcome To AANNG Sim Swap Fraud Alert Application!",  background='#ffffff', foreground='black', image=photo, compound='left',  font=("Helvetica", 16, "bold")  )  greeting2 = tk.Label(  text="How Would You Like To Proceed?",  background='#ffffff', foreground='black',  font=("Helvetica", 14, "bold")  )  greeting.pack(pady=10)  greeting2.pack(pady=5)  greeting.image = photo  # shabodi icon on the window  path = "C:/Users/Neyssa/Downloads/shabodi\_icon.jpg"  load = Image.open(path)  render = ImageTk.PhotoImage(load)  window.iconphoto(False, render)  # load and decrypt ICCID data from JSON file  def load\_and\_decrypt\_data(file\_path):  try:  with open(file\_path, 'rb') as encrypted\_file:  encrypted\_data = encrypted\_file.read()  # Load encrypted data and decrypt it  # cipher = Fernet(key)  # decrypted\_data = cipher.decrypt(encrypted\_data).decode()  # return np.array(json.loads(decrypted\_data))  except Exception as e:  print(f"Error loading encrypted ICCIDs: {e}")  return np.array([])  # bandwidth choking for a fraudulent SIM using API  def limit\_bandwidth\_for\_sim(iccid):  url = 'http://127.0.0.1:7999/qos/v1/bandwidth' # bandwidth API endpoint  data = {  "iccid": iccid, # suspicious SIM's ICCID  "action": "limit", # limit bandwidth  "limit": "128kbps" # set a lower bandwidth limit (arbitrary)  }  # send API request to limit bandwidth  try:  response = requests.post(url, json=data)  response\_data = response.json()  if response.status\_code == 200 and response\_data.get("status") == "success":  messagebox.showinfo("Success", f"Bandwidth successfully limited for SIM: {iccid}")  else:  messagebox.showerror("Error", f"Failed to limit bandwidth for SIM: {iccid}")  except requests.exceptions.RequestException as e:  print(f"Error limiting bandwidth: {e}")  messagebox.showerror("Error", "An error occurred while limiting bandwidth.")  def check\_latency(iccid):  url = f'http://127.0.0.1:7999/latency/v1/check/{iccid}' # latency API endpoint  try:  response = requests.get(url)  if response.status\_code == 200:  latency\_data = response.json()  latency = latency\_data.get("latency")  if latency:  messagebox.showinfo("Latency Check", f"Latency for SIM {iccid}: {latency} ms")  else:  messagebox.showwarning("Latency Check", f"No latency data available for SIM {iccid}.")  else:  messagebox.showerror("Error", "Failed to check latency.")  except requests.exceptions.RequestException as e:  print(f"Error checking latency: {e}")  messagebox.showerror("Error", "An error occurred while checking latency.")  # detect fraudulent activity (unregistered or location-swapped SIM cards)  def detection(current\_iccid\_data, registered\_iccids, window):  detected\_issues = []  for entry in current\_iccid\_data:  # get SIM number and location of SIM  iccid = entry.get('iccid')  location = entry.get('location')  if iccid not in registered\_iccids:  detected\_issues.append(f"Unregistered SIM detected in network: {iccid}, {location}")  else:  registered\_location = registered\_iccids.get(iccid)  if location != registered\_location:  detected\_issues.append(f"Location change detected for SIM {iccid}: {location}")  # alert mechanism  if detected\_issues:  if messagebox.askyesno(  "Fraudulent Activity Detected",  "\n".join(detected\_issues) + "\nDo you want to accept the SIM swap?"  ):  iccid\_to\_swap = messagebox.askstring("Accept SIM Swap", "Enter ICCID to accept SIM swap:")  if iccid\_to\_swap:  process\_sim\_swap\_acceptance(iccid\_to\_swap)  else:  # bandwidth choking  if messagebox.askyesno("Limit Bandwidth", "Do you want to limit the bandwidth for this SIM?"):  iccid\_to\_limit = messagebox.askstring("Limit Bandwidth", "Enter ICCID to limit bandwidth:")  if iccid\_to\_limit:  limit\_bandwidth\_for\_sim(iccid\_to\_limit)  if messagebox.askyesno("Check Latency", "Do you want to check latency for suspicious SIM?"):  iccid\_to\_check\_latency = messagebox.askstring("Check Latency", "Enter ICCID to check latency:")  if iccid\_to\_check\_latency:  check\_latency(iccid\_to\_check\_latency)  # SIM Swap acceptance  def process\_sim\_swap\_acceptance(iccid):  try:  log\_message = f"SIM swap accepted for ICCID: {iccid}"  print(log\_message)  with open("sim\_swap\_log.txt", "a") as log\_file:  log\_file.write(f"{log\_message}\n")  messagebox.showinfo("SIM Swap Accepted", f"SIM swap successfully accepted for ICCID: {iccid}")  except Exception as e:  messagebox.showerror("Error", f"Failed to process swap: {e}")  # load JSON data and display map  def view\_iccid\_map():  # check that the JSON file exists in the same directory as the script  json\_file\_path = 'sample\_json\_callback.json'  if not os.path.exists(json\_file\_path):  messagebox.showerror("File Error", "The JSON file was not found.")  return  # load ICCID and coordinates from JSON file  with open(json\_file\_path, 'r') as file:  data = json.load(file)  # initialize map (centered on an average location or a specific point)  map\_center = [37.7749, -122.4194] # ex. center (San Francisco)  map\_object = folium.Map(location=map\_center, zoom\_start=5)  # add markers for each ICCID location  for item in data:  iccid = item["ICCID"]  lat = item["location"]["latitude"]  lon = item["location"]["longitude"]  folium.Marker(  location=[lat, lon],  popup=f"ICCID: {iccid}",  tooltip="Click for more info"  ).add\_to(map\_object)  # save map to HTML  map\_file = 'map.html'  map\_object.save(map\_file)  # open the map in the default web browser  webbrowser.open(f'file://{os.path.realpath(map\_file)}')  # window for monitoring sim card activity  def open\_new\_window():  new\_window = tk.Toplevel(window)  new\_window.title("Monitoring Sim Card Activity")  new\_window.geometry("400x300")  json\_file\_path = 'sample\_json\_callback.json'  # check if the JSON file exists  if not os.path.exists(json\_file\_path):  messagebox.showerror("File Error", "The JSON file with SIM data was not found.")  return  # load and display JSON data  with open(json\_file\_path, 'r') as file:  sim\_data = json.load(file)  # create a text widget to display the JSON data in new window  text\_widget = tk.Text(new\_window, wrap='word', font=("Helvetica", 10))  text\_widget.pack(expand=True, fill='both')  # insert formatted JSON data into text widget  formatted\_data = json.dumps(sim\_data, indent=4)  text\_widget.insert(tk.END, formatted\_data)  text\_widget.config(state=tk.DISABLED) # make the text widget read-only  # new window for sim card history  def open\_new\_window2():  new\_window2 = tk.Toplevel(window)  new\_window2.title("Sim Card History")  new\_window2.geometry("400x300")  tk.Label(new\_window2, text="Sim Card History", font=("Helvetica", 14, "bold")).pack(pady=10)  history\_label = tk.Label(new\_window2, text="History details will be shown here.", font=("Helvetica", 12))  history\_label.pack(pady=10)  # Settings window  def open\_settings\_window():  settings\_window = tk.Toplevel(window)  settings\_window.title("Settings")  settings\_window.geometry("400x500")  tk.Label(settings\_window, text="Settings", font=("Helvetica", 14, "bold")).pack(pady=20)  # latency API button  latency\_button = tk.Button(  settings\_window, text="Check Latency API", command=open\_latency\_window,  height=2, width=25, background='#ffffff', foreground='black', font=("Helvetica", 12, "bold")  )  latency\_button.pack(pady=10)  # bandwidth API button  bandwidth\_button = tk.Button(  settings\_window, text="Check Bandwidth API", command=open\_bandwidth\_window,  height=2, width=25, background='#ffffff', foreground='black', font=("Helvetica", 12, "bold")  )  bandwidth\_button.pack(pady=10)  # SIM swap API button  sim\_swap\_button = tk.Button(  settings\_window, text="Check SIM Swap API", command=open\_sim\_swap\_window,  height=2, width=25, background='#ffffff', foreground='black', font=("Helvetica", 12, "bold")  )  sim\_swap\_button.pack(pady=10)  # location API button  location\_button = tk.Button(  settings\_window, text="Check Location API", command=open\_location\_window,  height=2, width=25, background='#ffffff', foreground='black', font=("Helvetica", 12, "bold")  )  location\_button.pack(pady=10)  def open\_latency\_window():  latency\_window = tk.Toplevel(window)  latency\_window.title("Latency API Information")  latency\_window.geometry("400x300")  tk.Label(latency\_window, text="Latency API Information", font=("Helvetica", 14, "bold")).pack(pady=20)  iccid\_input = simpledialog.askstring("Latency Check", "Enter ICCID to check latency:")  if iccid\_input:  check\_latency(iccid\_input)  def open\_bandwidth\_window():  bandwidth\_window = tk.Toplevel(window)  bandwidth\_window.title("Bandwidth API Information")  bandwidth\_window.geometry("400x300")  tk.Label(bandwidth\_window, text="Bandwidth API Information", font=("Helvetica", 14, "bold")).pack(pady=20)  iccid\_input = simpledialog.askstring("Bandwidth Check", "Enter ICCID to limit bandwidth:")  if iccid\_input:  limit\_bandwidth\_for\_sim(iccid\_input)  def open\_sim\_swap\_window():  sim\_swap\_window = tk.Toplevel(window)  sim\_swap\_window.title("SIM Swap API Information")  sim\_swap\_window.geometry("400x300")  tk.Label(sim\_swap\_window, text="SIM Swap API Information", font=("Helvetica", 14, "bold")).pack(pady=20)  iccid\_input = simpledialog.askstring("SIM Swap Check", "Enter ICCID to check swap status:")  if iccid\_input:  process\_sim\_swap\_acceptance(iccid\_input)  def open\_location\_window():  location\_window = tk.Toplevel(window)  location\_window.title("Location API Information")  location\_window.geometry("400x300")  tk.Label(location\_window, text="Location API Information", font=("Helvetica", 14, "bold")).pack(pady=20)  # user interface  monitor\_button = tk.Button(  window, text="Monitor Sim Activity", command=open\_new\_window,  height=3, width=25, background='#ffffff', foreground='black',  font=("Helvetica", 14, "bold")  )  monitor\_button.pack(pady=20)  history\_button = tk.Button(  window, text="Sim Card History", command=open\_new\_window2,  height=3, width=25, background='#ffffff', foreground='black',  font=("Helvetica", 14, "bold")  )  history\_button.pack(pady=20)  view\_iccid\_map\_button = tk.Button(  window, text="View ICCID Map", command=view\_iccid\_map,  height=3, width=25, background='#ffffff', foreground='black',  font=("Helvetica", 14, "bold")  )  view\_iccid\_map\_button.pack(pady=20)  settings\_button = tk.Button(  window, text="Settings", command=open\_settings\_window,  height=3, width=25, background='#ffffff', foreground='black',  font=("Helvetica", 14, "bold")  )  settings\_button.pack(pady=20)  exit\_button = tk.Button(window, text="Exit", command=window.quit, height=3, width=25, background='#ffffff', foreground='black',  font=("Helvetica", 14, "bold"))  exit\_button.pack(pady=10)  window.mainloop() |
| --- |

Direct code from file SIM\_swap\_legitimacy.py:

| import json  import numpy as np  import time  import requests  from datetime import datetime  from cryptography.fernet import Fernet  # initialize JSON file containing the registered SIMs (assume provided by the network)  json\_file\_path = 'registered\_sims.json'  # generate key  try:  with open('encryption\_key.key', 'rb') as key\_file:  key = key\_file.read()  except FileNotFoundError:  key = Fernet.generate\_key()  with open('encryption\_key.key', 'wb') as key\_file:  key\_file.write(key)  cipher = Fernet(key)  # encrypt data and save to file  def encrypt\_and\_save\_data(data, file\_path):  json\_data = json.dumps(data) # convert data to JSON string  encrypted\_data = cipher.encrypt(json\_data.encode()) # encrypt JSON string  with open(file\_path, 'wb') as encrypted\_file:  encrypted\_file.write(encrypted\_data)  # decrypt and load data from file  def load\_and\_decrypt\_data(file\_path):  try:  with open(file\_path, 'rb') as encrypted\_file:  encrypted\_data = encrypted\_file.read()  decrypted\_data = cipher.decrypt(encrypted\_data).decode()  # return np.array(json.loads(decrypted\_data))  except Exception as e:  print(f"Error loading encrypted ICCIDs: {e}")  return np.array([])  # configure latency API  LATENCY\_API\_URL = "http://127.0.0.1:7999/qos/v1/latency"  # set device latency  def set\_device\_latency(device\_id, latency, duration):  try:  data = {  "device": {  "deviceId": device\_id  },  "latency": latency,  "duration": duration  }  response = requests.post(LATENCY\_API\_URL, json=data)  if response.status\_code == 201:  response\_data = response.json()  transaction\_id = response\_data.get("transactionId")  remaining\_time = response\_data.get("remainingTime")  print(f"Latency set successfully. Transaction ID: {transaction\_id}, Remaining time: {remaining\_time}")  return transaction\_id, remaining\_time  else:  print(f"Failed to set latency: {response.status\_code} - {response.json()}")  return None, None  except requests.exceptions.RequestException as e:  print(f"Error setting latency: {e}")  return None, None  # get latency session information  def get\_device\_latency\_sessions(device\_id):  try:  response = requests.get(f"{LATENCY\_API\_URL}?deviceId={device\_id}")  if response.status\_code == 200:  latency\_data = response.json()  return latency\_data.get('sessions', [])  else:  print(f"Failed to retrieve latency sessions: {response.status\_code} - {response.json()}")  return []  except requests.exceptions.RequestException as e:  print(f"Error fetching latency sessions: {e}")  return []  # put ICCIDs into an array by parsing JSON file  def load\_iccids(file\_path):  return load\_and\_decrypt\_data(file\_path)  # configure bandwidth API  API\_URL = "http://127.0.0.1:7999/qos/v1/bandwidth"  LOCATION\_API\_URL = "http://127.0.0.1:7999/location/v1/center"  DEVICE\_ID = "17" # this is specific to our group based on the email they gave  AEP\_HOST = "192.168.3.18" # Shabodi server IP  CLIENT\_ID = "380dfca1-6539-4890-9595-21a57c8f907d"  CLIENT\_SECRET = "enMKQcRauITqqdsDGsbNDUN\_JlNgLYQkkdPhe5IF8Ws"  def get\_token(client\_id, client\_secret):  url = f"http://{AEP\_HOST}:31002/security/v1/token"  headers = {  "Content-Type": "application/json"  }  data = {  "client\_id": client\_id,  "client\_secret": client\_secret  }  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  # fetch the device's location  def fetch\_device\_location(token, device\_id):  try:  headers = {  "Authorization": f"Bearer {token}",  "accept": "application/json"  }  response = requests.get(f"{LOCATION\_API\_URL}?deviceId={device\_id}", headers=headers)  if response.status\_code == 200:  location\_data = response.json()  if "location" in location\_data:  device\_location = location\_data["location"][0]  return device\_location # return device latitude, longitude, and altitude  else:  print(f"Failed to retrieve location: {response.status\_code}, {response.json()}")  return None  else:  print(f"Failed to fetch device location: {response.status\_code}")  return None  except requests.exceptions.RequestException as e:  print(f"Error fetching location: {e}")  return None  # verify the location within a specific area  def verify\_device\_location(token, device\_id, latitude, longitude, radius):  try:  headers = {  "Authorization": f"Bearer {token}",  "accept": "application/json"  }  body = {  "device": {  "deviceId": device\_id  },  "area": {  "areaType": "CIRCLE",  "center": {  "latitude": latitude,  "longitude": longitude  },  "radius": radius  },  "maxAge": 120  }  response = requests.post(f"{LOCATION\_API\_URL}/verify", headers=headers, json=body)  if response.status\_code == 200:  verification\_result = response.json()  return verification\_result  else:  print(f"Failed to verify device location: {response.status\_code}, {response.json()}")  return None  except requests.exceptions.RequestException as e:  print(f"Error verifying location: {e}")  return None  # log data from bandwidth  # we can potentially make a graph from this, but it isn't necessary for our MVP  def log\_bandwidth\_data(data):  data['timestamp'] = datetime.now().isoformat()  with open('bandwidth\_log.json', 'a') as log\_file:  log\_file.write(json.dumps(data) + "\n")  # detect swap or duplication  def detection(current\_iccid\_data, registered\_iccids):  detected\_issues = []  for entry in current\_iccid\_data:  iccid = entry.get('iccid')  location = entry.get('location')  if iccid not in registered\_iccids:  detected\_issues.append(f"Unregistered SIM detected in network: {iccid}, {iccid}: {location}")  else:  registered\_location = registered\_iccids.get(iccid)  if location != registered\_location:  detected\_issues.append(f"Location change detected for SIM {iccid}: {location}")  if detected\_issues:  print("Detected SIM issues:", detected\_issues)  if \_\_name\_\_ == "\_\_main\_\_":  token = get\_token(CLIENT\_ID, CLIENT\_SECRET)  if token:  while True:  registered\_iccids = load\_iccids(json\_file\_path)  print("Registered ICCIDs:", registered\_iccids)  # fetch current data with token  bandwidth\_data = fetch\_bandwidth\_data(token)  if bandwidth\_data:  log\_bandwidth\_data(bandwidth\_data)  print("Logged bandwidth data:", bandwidth\_data)  if 'sessions' in bandwidth\_data:  detection(bandwidth\_data['sessions'], registered\_iccids)  device\_location = fetch\_device\_location(token, DEVICE\_ID)  if device\_location:  print(f"Device {DEVICE\_ID} location: Latitude={device\_location['latitude']}, Longitude={device\_location['longitude']}, Altitude={device\_location['altitude']}")  center\_latitude = 50.735851  center\_longitude = 7.10066  radius = 50000 # 50 km radius  location\_verification = verify\_device\_location(token, DEVICE\_ID, center\_latitude, center\_longitude, radius)  if location\_verification:  print(f"Location verification result: {location\_verification['verificationResult']}")  time.sleep(3600) # hourly loop |
| --- |

Direct (snippet) copy of sample JSON file used for testing purposes, sample\_JSON\_callback.json:

* Please note: The entire file cannot be included, since it includes over 7000 lines of data.

| {  "ICCID": "88506093933586154906",  "location": {  "latitude": -60.1494,  "longitude": 130.2527  }  },  {  "ICCID": "97462394770174064702",  "location": {  "latitude": -79.6684,  "longitude": 128.9968  }  },  {  "ICCID": "16863611598075024893",  "location": {  "latitude": -80.2355,  "longitude": 89.9004  }  },  {  "ICCID": "74069550433877912184",  "location": {  "latitude": 72.3902,  "longitude": 155.4221  }  },  {  "ICCID": "87139085241161508679",  "location": {  "latitude": -87.5375,  "longitude": -46.8326  }  },  {  "ICCID": "48704783487768753522",  "location": {  "latitude": 40.6462,  "longitude": -126.9088  }  },  {  "ICCID": "18278952873096762065",  "location": {  "latitude": 46.767,  "longitude": -166.3167  }  },  {  "ICCID": "10091781954490649479",  "location": {  "latitude": -89.3549,  "longitude": -57.1879  }  },  {  "ICCID": "96388837294680634566",  "location": {  "latitude": -4.9899,  "longitude": 108.4714  }  },  {  "ICCID": "92700701495392143188",  "location": {  "latitude": -60.2909,  "longitude": 174.1122  }  }  ] |
| --- |