To: Professor Pisano

From: Beren Donmez, Margherita Piana, Marissa Ruiz, Bennet Taylor, Albert Zhao,

Team: 8

Date: 11/21/24

Subject: Bike Guard First Prototype Test Report

### **1.0 Equipment and Setup**

* **Hardware**
  + Raspberry Pi Zero 2 W with 32GB SanDisk SDHC Class 10 card
  + Piezo Buzzer
  + MPU-6050 Accelerometer
  + INIU BI-B61 Portable Charger (22.5W, 10000mAh)
  + Raspberry Pi Camera Module V2
  + TP-Link Router
  + Small breadboard
  + Electrical Tape
  + Jumper Cables
  + USB-C to Micro USB Cable
  + Small heat sinks
* **Remote Equipment**
  + Laptop
* **Software**
  + Raspberry Pi OS 32-bit (Legacy) – Debian Bullseye
  + Node.js – for child\_process and file system operations
  + Python3 – accelerometer data reading
  + Flask – to stream camera feed and handle back-end tasks
  + Front-end: JavaScript, React, and CSS
  + Back-end: Flask, SQL for data storage
  + Machine Learning Model: Logistic Regression
* **Setup**
  + Hardware components placed in a lockbox mounted to the bike, Raspberry Pi Camera peaks out of the enclosure to capture video data
  + Raspberry Pi and all servers are connected to TP-Link Router
  + Raspberry Pi is powered by a portable power bank
  + Small heat sinks are attached to Raspberry Pi’s onboard chip components
  + Raspberry Pi connected via SSH using ssh Team8@Raspberry\_pi\_IP
  + Accelerometer data collected and saved in CSV files using Node.js and Python scripts (for later use by machine learning model, not currently incorporated)
  + Buzzer connected to Raspberry Pi, goes off when Pitch and Roll values reach hardcoded threshold (in future will be handled by machine learning model)
  + Pi Camera data is taken using Python and streamed to local host using Python and Flask
  + When buzzer goes off, push request is sent to back-end and stored in an SQL database
  + Front-end monitors for changes in back-end, uses React to update changes on web interface (npm start)
  + Front-end embeds camera stream from Raspberry Pi’s local host to the web interface (python script to connect front-end and back-end)

### **2.0 Measurements Taken**

#### **2.1 Hardware and Connectivity**

* Verified that the Raspberry Pi boots up correctly with the portable power bank by checking that green LED on the Raspberry Pi turns on
* Pinged Raspberry Pi after boot to ensure network connection to router was successful
* Confirmed successful communication between peripherals (accelerometer, buzzer) and Raspberry Pi by running our accelerometer.py independently from other processes
* Confirmed camera stream was successful and sent to local host by running picam3.py independently of other processes

#### 

#### **2.2 Data Recording and Alerts**

* Tested accelerometer data logging with Node.js and confirmed that motion is recorded in CSV format
* Verified the bike triggers the buzzer when shaked and sends a message to the front-end
* Observed that the website updates in real-time when the accelerometer detects the shaking
* Observed real-time camera stream on website

#### **2.3 Machine Learning and Classification**

* Trained the logistic regression model on pre-collected accelerometer data
* Assessed model performance using the confusion matrix:
  + True Positives: 49
  + False Positives: 3
  + False Negatives: 6
  + True Negatives: 18
* Calculated accuracy as approximately 88.16%

**3.0 Conclusions based on test data**

* The hardware setup functions as expected: the enclosure does not interfere with the bike’s operation, and all components operate reliably.
* The accelerometer successfully detects excessive shaking, triggers the buzzer, and logs data.
* The Raspberry Pi Camera shows a live video feed to the front-end without interruptions.
* The machine learning model achieves good performance, effectively identifying theft scenarios with high accuracy.
* The system meets the measurable criteria for a successful run: real-time alerts, immediate buzzer response, and reliable classification of motion.

**4.0 Extra Credit: Our Ideas for the Future**

* **Integrate Machine Learning Model to the Product:**Incorporate the trained logistic regression model directly into the Raspberry Pi system, enabling real-time detection and classification of theft scenarios without external dependencies
* **Incorporation of a Prepaid SIM Card:**Equip the system with a prepaid SIM card to enable data transmission over cellular networks, ensuring functionality even when Wi-Fi is unavailable
* **GPS Tracking Through the SIM Card:**Utilize the SIM card for GPS tracking, allowing the owner to monitor the bike's location remotely in case of theft or unauthorized movement.
* **Design and Print our Own Enclosure:**As it stands, our current enclosure feels too bulky and generalized. In the future we hope to downsize the enclosure and make it blend in with the bike, perhaps disguised as a bike light or other bike component. The smaller we can make it the less noticeable by thieves.
* **Move from web application to mobile application:**For our demo we showed live alerts and camera stream on a locally hosted website, we hope to move that to a mobile application
* **Switch to replaceable batteries:**For the sake of the demo, we used a power bank to power our device and make it mobile. In the future, we hope to assess our power budget and move to replaceable batteries
* **Linking user account and device via Bluetooth**During user setup, user can use Bluetooth to link their phone to their device