

GNG5140
Design Project User and Product Manual

Cycling Facility Surface Smoothness Logger

Submitted by:

SmoothRiders

Jose Sarmiento Tejada, 300363449

Jiaqing Zheng, 300441923

Sai Aditya Bommakanti, 300279989

10 December 2024

University of Ottawa

Table of Contents

Table of Contents	ii
List of Figures	iv
List of Tables	v
List of Acronyms and Glossary	vi
1 Introduction.....	1
2 Overview.....	2
2.1 System	3
2.2 Cautions & Warnings.....	6
2.2.1 User.....	6
2.2.2 Developer.....	6
3 Getting Started	8
3.1 System Organization & Navigation	8
4 Using the System	9
4.1 Menu Item – Menu Button.....	9
4.1.1 Menu Item – Read Backend File	9
4.1.2 Menu Item – Plot GPS History	9
4.1.3 Menu Item – Quit App.....	10
4.2 Menu Item – Upload Button.....	10
4.3 Text View Box	10
4.4 Map Element	10

5	Troubleshooting & Support	11
5.1	Error Messages or Behaviors	11
5.2	Special Considerations and Maintenance.....	13
5.3	Support	13
6	Product Documentation	14
6.1	Planned System Architecture	17
6.2	Bill of Materials (BOM).....	17
6.3	Testing & Validation.....	18
6.3.1	Performed V&V	21
7	Conclusions and Recommendations for Future Work	26
7.1	Lessons Learned.....	26
7.2	Planned Work.....	26
8	Bibliography	28
	APPENDIX.....	30

List of Figures

Figure 1: High-Level Architecture of the System.....	3
Figure 2: The Hardware Prototype	3
Figure 3: The Software Prototype.....	4
Figure 4: Flowchart of the Mobile Application	5
Figure 5: Mobile App Menu Options.....	8
Figure 6: Menu Options in the Application	9
Figure 7: GPS Location Markers	10
Figure 8: Example of a Location Error	11
Figure 9: Example of a Non-Implementation Error.....	11
Figure 10: An Example Wherein a Click on any other Part of the App Leads to the Notification Going Away.....	12
Figure 11: Designed System Architecture of the Complete Prototype.....	17
Figure 12: Example of adding functional and non-functional requirements to the system model.....	21
Figure 13: Example of requirement captured in the model	22
Figure 14: Verification performed in the architecture	22

List of Tables

Table 1: Acronyms.....	vi
Table 2: Solution Comparison Table	14
Table 3: The Planned Stages of Implementation	16
Table 4: Bill of Materials	18
Table 5: Verification and Validation Strategy	21
Table 6: Verification results.....	25
Table 7: Reference Resources	30

List of Acronyms and Glossary

Acronym	Definition
UPM	User and Product Manual
API	Application Programming Interface
PII	Personally Identifiable Information
UI	User Interface
FSR	Functional System Requirement
NFSR	Non-Functional System Requirement

Table 1: Acronyms

1 Introduction

This User and Product Manual (UPM) provides the information necessary for bikers and authorities of the municipalities of Ontario to effectively use the surface mapper solution and for prototype documentation. The solution and all related documentation are deliverables of the graduate level course GNG 5140 Engineering Design, Fall 2024 at the University of Ottawa. This document provides enough information for the use, maintenance, and improvement of the product as it presently exists.

Dedicated bike paths are a healthy and environmentally friendly alternative to fossil fuel powered vehicles. Several people use these paths daily for various reasons at different hours of the day. However, separated protected cycling facilities (dedicated bike lanes) may not be utilized by cyclists if the quality of construction is poor; specifically, the smoothness of the road. Poor smoothness forces the cyclists to use adjacent roadways leading to mixed traffic. In some cases, the poor quality of the bike path may even force the user to use other modes of transportation. Best practices for testing surface smoothness on cycling facilities need to be identified to maintain good quality cycling infrastructure. This project focused on developing a method to map the smoothness of a surface to provide reliable bicycle infrastructure data to both the cyclists and municipalities of Ontario.

The various sections in this document provide information to the reader on the background of the problem and solution, cautions, setup, using the system, troubleshooting, support and project documentation. The project report and all related files are maintained in a repository on MakerRepo. The Appendix contains more information on this.

2 Overview

The problem addressed by the project is two-fold. One is the problem that municipal authorities in Ontario face in evaluating the quality of bikeways and the other is the discomfort that bikers face when a biking surface is of bad quality. At the time of working on the solution, the team was unable to find standard testing methods and quality metrics to assess the quality – surface smoothness and anomalies especially – of a constructed biking surface. The solution tackles both the problems and provides additional, indirect benefits such as enhanced routing options in maps applications for bikers and so on.

The user in this case can either be a biker consenting to using their mobile phone for recording acceleration and GPS data, or any official from a municipality in Ontario collecting road test data.

The solution is novel, in the sense that there exists no solution resembling the system configuration such as this solution. While existing literature on the topic provides good quality information, it is research data, and no practical application has been identified by the team. In other words, there is no existing product that addresses the problem. However, the research data was taken into consideration by the team to produce the solution. The Bibliography section contains all references, including those from the project report.

2.1 System

The following images of the prototype give a brief idea into the design of the system.

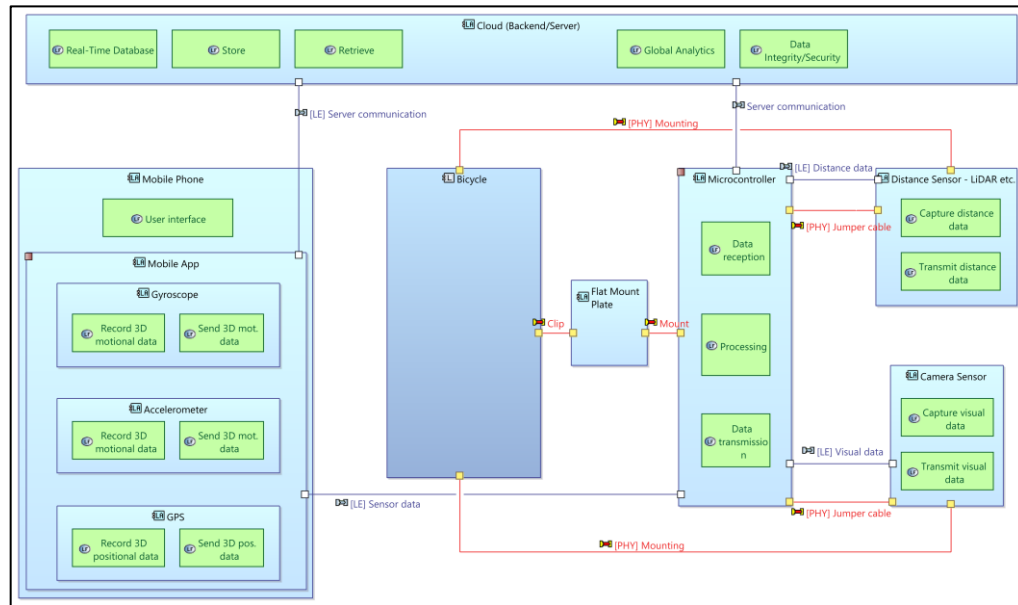


Figure 1: High-Level Architecture of the System

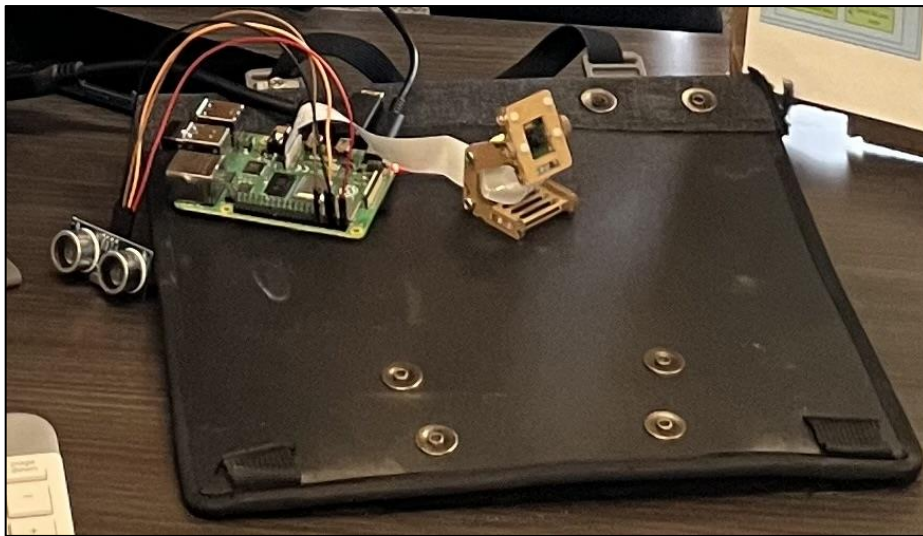


Figure 2: The Hardware Prototype

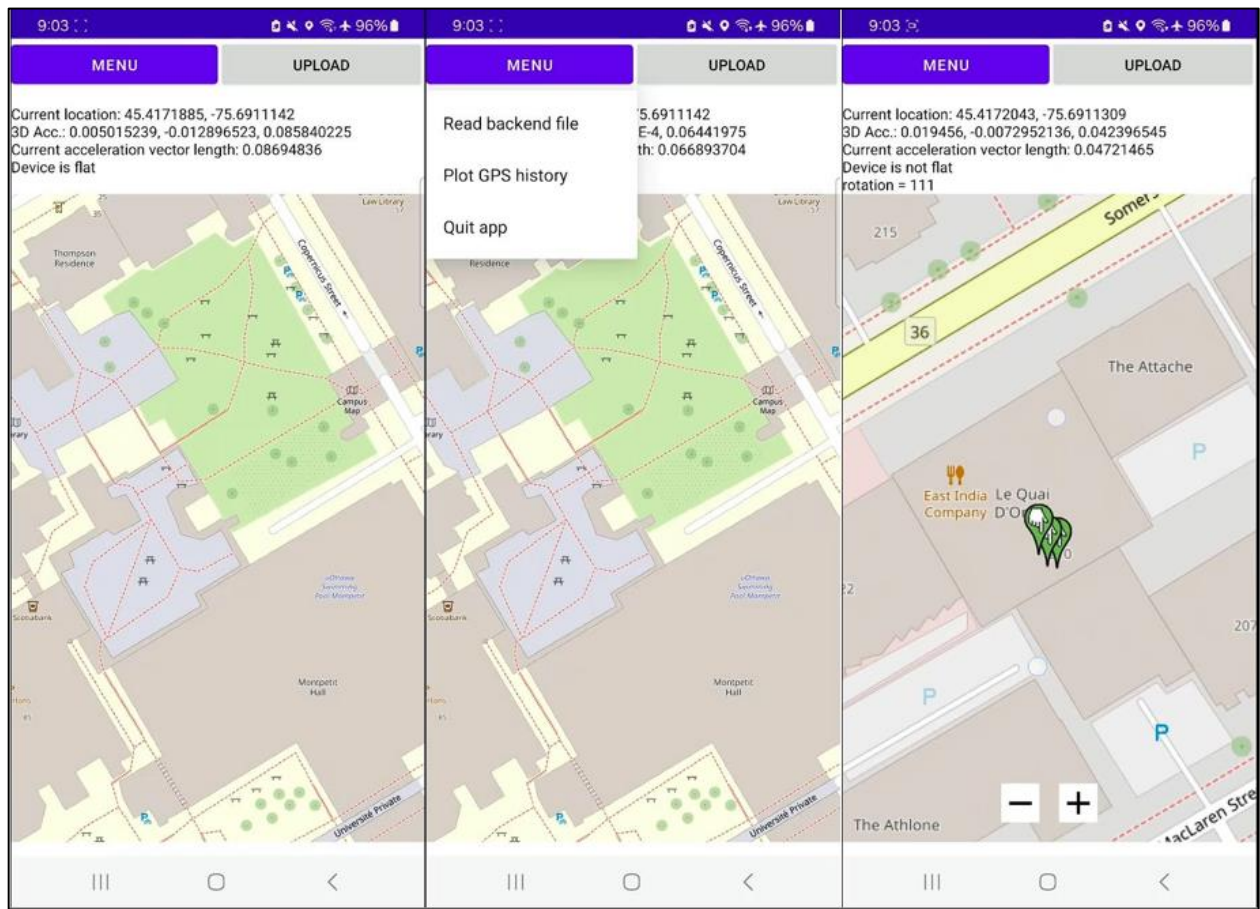


Figure 3: The Software Prototype

As can be seen from the pictures, the prototype (solution) is two-pronged; it consists of a hardware setup and a software setup. The complete solution has the following breakup of components.

- Hardware setup – A mobile phone (with accelerometer and GPS sensors), a microcontroller, ultrasound sensors, camera sensor
- Software setup – Android mobile application and microcontroller code

The system was designed to be modular. Even without the dedicated hardware – external sensors and microcontroller – the system can still function and provide the required information.

The prototype makes several assumptions, two of the most important ones being that the user has an Android mobile phone on them and that their bike has space for a back-plate or basket to snap on to. Additionally, the user must provide their consent to record their positional and motional sensor readings for the solution to work. The following software (logical) flowchart provides more information on the basic functionality of the prototype.

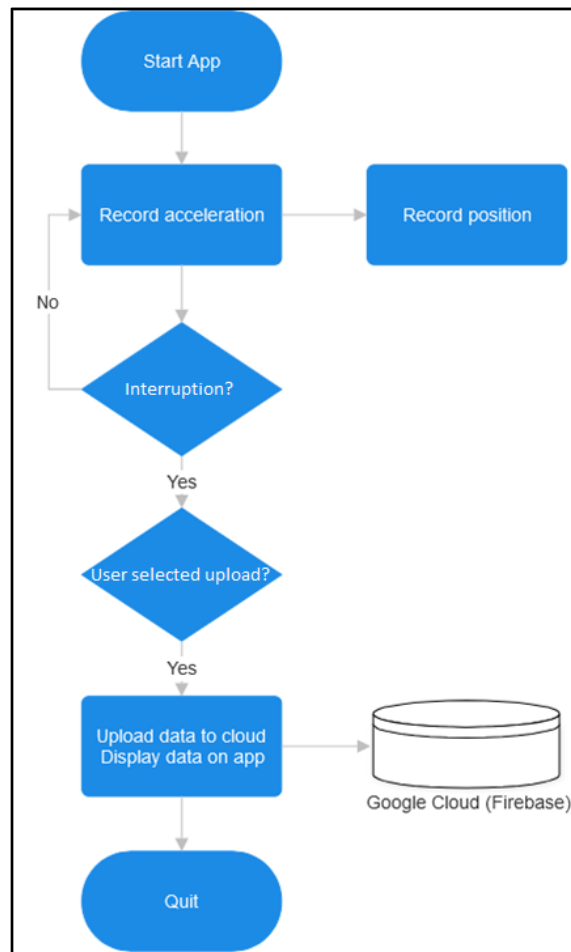


Figure 4: Flowchart of the Mobile Application

2.2 Cautions & Warnings

2.2.1 User

The user must provide the required permissions in the application and understand that by using the solution they consent to their positional and motional data, including potentially personally identifiable information (PII) that sometimes may be in the form of an image, being recorded and stored on a server that may or may not be owned and maintained by the developer or the client.

2.2.2 Developer

The cloud service used for this prototype is Google Cloud (through Google Firebase). While it provides limited free usage, the service required the team to input their credit card details during setup. As a result, the code and all related information included in the MakerRepo files of this project may not work until a new cloud service is created/selected and appropriately integrated with the software.

The team recommends Google Firebase since the prototype was based on it and uses some Application Programming Interfaces (APIs) provided by them; this would require the least amount of edits in the code files.

3 Getting started

The following steps cover the general setup and operational information of the prototype. Not all steps need to be performed every time the solution is used, but it is recommended to verify every step before every use of the system.

These steps cover the end-to-end operational setup of the solution. If the user wishes to not use certain parts of the solution, for example, the hardware, they can skip those steps. The modularity of the solution is in the hands of the user.

1. Ensure that the mobile app is installed.
2. Ensure that all permissions requested by the application are provided.
3. Open any maps application on the phone at least once and verify that the GPS sensor is working.

Note: This is needed due to the way the app is coded. A power-saving location API was used which unfortunately does not “force” the phone to fetch new location data. As a result, the GPS sensor needs to be “woken up” by other means before the solution can take the required control over the sensor.

4. Ensure that the hardware is secured on a flat surface with the following orientation of the sensors.
 - a. Two ultrasound sensors covering the lateral plane – one facing left and the other facing right.
 - b. A camera facing the road surface. (Another camera facing sideways and/or upwards may be used).
5. Ensure that the flat surface is properly mounted on to the bike either at the front or at the back.
6. Ensure that the mobile phone is laid reasonably flat – the same surface used to mount the sensors can be utilized to hold the mobile phone.

7. Run the mobile application and power on the microcontroller.

The user does not need to do anything else other than go about their usual activities. They are only required to close the mobile application and switch off the microcontroller once they reach their destination or wish to stop using the solution.

Note: The implementation of the solution is not complete. The project report explains further and contains the complete strategy and subsequent plans. The hardware and software integration – microcontroller and mobile app – is not complete. As a result, the bulk of this user manual focuses on the mobile application. Any work performed on the actual system warrants a respective update of this document by the development team.

3.1 System Organization & Navigation

The following image captures the menu elements of the application. The terms are self explanatory, and the steps above cover all that the user needs to do to use this version of the prototype.

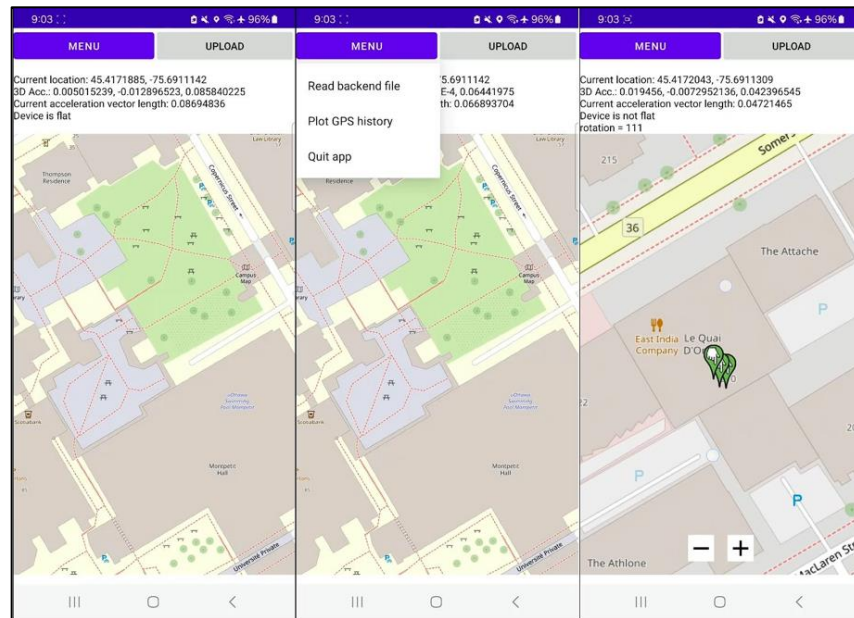


Figure 5: Mobile App Menu Options

4 Using the System

The software application is described in detail in this section. For the most part, no human intervention is needed, as the steps in the previous section highlight. The application only has one screen (activity).

4.1 Menu Item – Menu Button

The menu button is the button on the top left colored in purple. This button acts as the main menu button for the application.

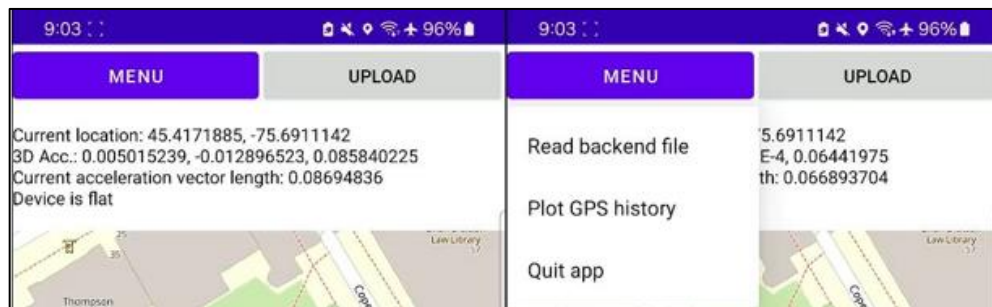


Figure 6: Menu Options in the Application

The following sub-functions are nested in the dropdown menu option.

4.1.1 Menu Item – Read Backend File

This feature has not yet been implemented.

Developer Note: The raw data can be retrieved from the server, but no UI element has been developed to present meaningful information to the user.

4.1.2 Menu Item – Plot GPS History

This button fetches all the GPS coordinates available in the file on the server and adds markers on the map in the application.



Figure 7: GPS Location Markers

4.1.3 Menu Item – Quit App

This button quits the application and returns the user to the screen that was open before the application was launched. The connection to the server is severed and no data is recorded.

4.2 Menu Item – Upload Button

The upload button uploads all recorded data – in the current session – to the server.

Developer Note: This overwrites all previous data in the existing file.

4.3 Text View Box

The text box is used to indicate the current readings to the user. It shows the current GPS coordinates and acceleration vector lengths.

4.4 Map Element

The interactive map shows the GPS location markers and allows the user to interact with it. The map is provided by OpenStreetMap and does not require a license to be purchased.

Developer Note: If the application is made commercial, OpenStreetMap regulations need to be thoroughly reviewed and abided by.

5 Troubleshooting & Support

The application software was designed to never let the application crash on the user. This was done by employing error handling techniques throughout the code. As a result, even in the event of a functional failure, the app simply displays a message to the user saying that the task could not be completed. The following sections provide an example of this behaviour.

5.1 Error Messages or Behaviors

The error messages in the application are in the form of Android toast messages – floating text typically at the bottom of the screen. The following images are two of the most common errors that may occur while using the application due to missing implementation or missing permissions.

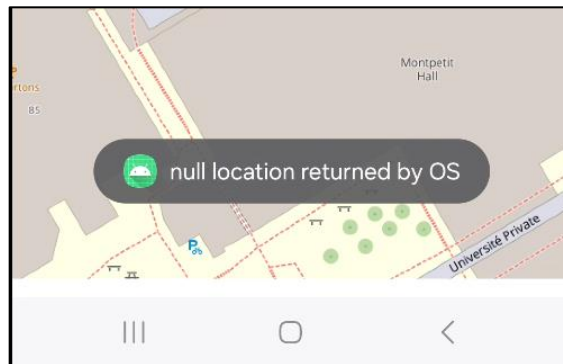


Figure 8: Example of a Location Error

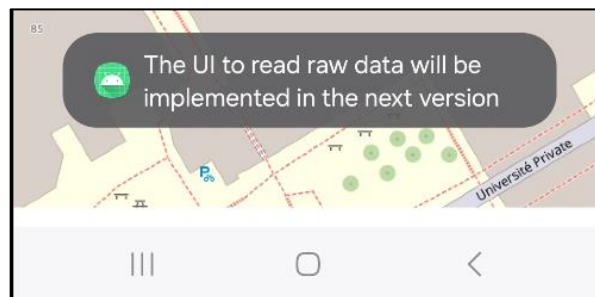


Figure 9: Example of a Non-Implementation Error

Other errors such as file not existing, file not able to be created, connection issues with the server and so on are handled but are not displayed to the user. These errors are completely abstracted away from the user. The only indication that the user will get of such errors is the long wait time, which the user can easily bypass by clicking on any part of the screen. An example is included below.

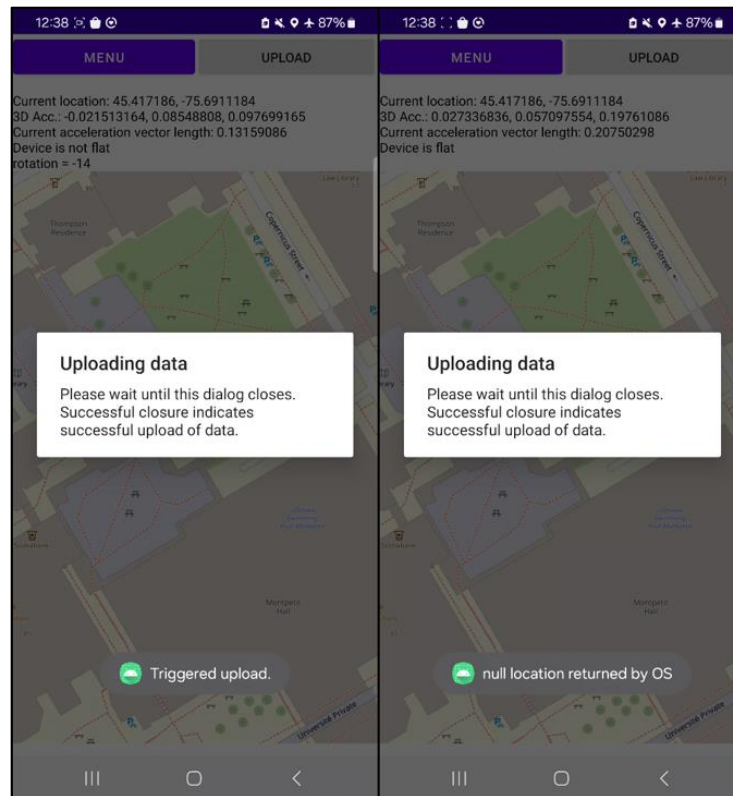


Figure 10: An Example Wherein a Click on any other Part of the App Leads to the Notification Going Away

5.2 Special Considerations and Maintenance

The user does not need to worry about any special considerations or application maintenance. Application updates may be made available by the developer at which point the user may wish to update the application.

Developer Note: The hardware part of the prototype is abstracted away from this manual due to no integration between the mobile application and the microcontroller. It should be a part of the next version of the prototype.

5.3 Support

All application support is conducted via email. Given the nature of the solution – a software application that poses no immediate threat to any life or property – there exists no emergency assistance. That being said, the team has 24x7 access to all the listed emails and will respond as soon as possible to the user.

For any kind of support or problem reporting, the user is requested to contact either one of the people from the development team included below.

1. Sai Aditya Bommakanti – sbomm094@uottawa.ca
2. Jiaqing Zheng – jzhen028@uottawa.ca
3. Jose Sarmiento Tejada – jsarm054@uottawa.ca

Important Note: For any (data) security incident or high severity issue that directly impacts the user, please make your subject line “HIGH SEVERITY”.

6 Product Documentation

The implementation of the prototype as it exists at the time of writing this report is included below.

The table below contains reasoning behind the selection of the hybrid solution strategy. This strategy is compared against a pure software (mobile application only) strategy and a state-of-the-art hardware solution strategy (dedicated sensors around the bike). Tick marks indicate the magnitude of advantage and crosses indicate the magnitude of disadvantage, both comparative.

Criteria	Concept 1: State-of-the-Art Sensors	Concept 2: Mobile Phone App	Solution: Mobile App + Sensors
High Accuracy *	✓✓✓	××	✓✓
Reliable Data *	✓✓✓	××	✓✓
Cost-Efficiency *	×	✓✓✓	✓✓
Compact/ Portable	×××	✓✓✓	✓
Development Time *	××	✓✓✓	✓
Modularity	✓✓✓	××	✓✓✓
Ease of Use	××	✓✓✓	✓✓
User Experience	×	✓✓✓	✓✓
Size Constraints	×××	✓✓✓	×
Cost *	×××	✓✓✓	×
Power Source Requirements	×××	✓✓✓	×
Data Privacy Concerns	✓✓✓	×××	×
Calibration Complexity	×××	✓✓✓	××
Development Complexity *	××	✓✓	✓
Weight	×××	✓✓✓	×
Platform Limitations	✓✓✓	×××	✓

Table 2: Solution Comparison Table

The solution was designed to be implemented over six stages, of which only four (stages 1 to 4) are done at the time of writing this manual. This is the reason why the microcontroller part of the solution is abstracted away from this user manual. The following table includes all six stages including the required equipment.

Stage	Elaboration
Stage 1	<p>An Android to exploit various standard sensors available on most smartphones. The Android platform was chosen because,</p> <ol style="list-style-type: none"> It is open source. Android has a global market share of 71.85%. An android phone is relatively inexpensive and provides more developer options out of the box. <p>The following sensors are tested/utilized.</p> <ol style="list-style-type: none"> Accelerometer (linear acceleration, 3 axes) – m/s^2 Gyroscope (3 axes) – rad/s Rotation Vector – unitless values along 3 axes GPS – unitless coordinates along 3 axes <p>Google Firebase was selected to provide a backend to the mobile application. Firebase was chosen because it easily integrates with Android code and provides a limited free-tier subscription model that is easy to setup.</p>
Stage 2	<p>The team will either identify a mathematical model on which to base the acquired data or develop a simple model or logic to derive meaningful information.</p>

Stage 3	<p>The following sensors were identified for version 1 of the application of the strategy.</p> <ol style="list-style-type: none"> 1. Distance sensors – LiDAR and/or ultrasound. 2. Visual sensors – Camera. 3. Microcontroller – Arduino or Raspberry Pi.
Stage 4	Raspberry Pi Python code will be written to power and control the identified sensors.
Stage 5	Interfacing the microcontroller and the mobile app may require additional sensor(s) such as Bluetooth or GSM.
Stage 6	Weatherproof Housing for the microcontroller and sensors would be designed for fit and 3D-printed. The housing would also include a self-sufficient power source such as a 12V rechargeable lead acid DC battery or equivalent (USB powered power bank).

Table 3: The Planned Stages of Implementation

6.1 Planned System Architecture

A model-based system engineering approach was selected for the project. Additionally, the design principle of Keep It Simple, Silly (KISS) was strictly followed.

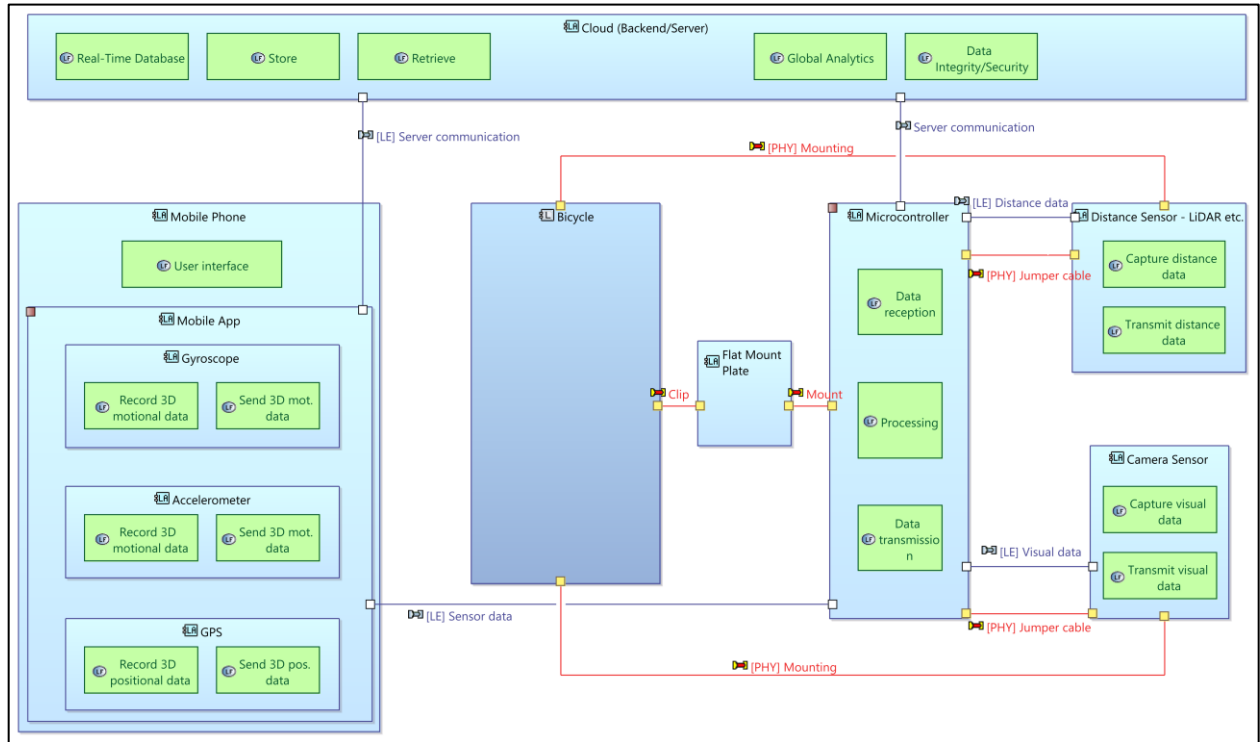


Figure 11: Designed System Architecture of the Complete Prototype

6.2 Bill of Materials (BOM)

The table below includes the bill of materials for the existing prototype.

#	Item	Qty.	Price (CAD)	Link/Remarks
1	Raspberry Pi 4 Model B-4GB	1	\$ 90.00	https://makerstore.ca/shop/ols/products/raspberry-pi-4-model-b-4gb

2	FNK0056 camera	1	\$ 13.99	https://a.co/d/gowOgW7
3	Power bank	1	\$ 19.55	https://a.co/d/gsBaxO3
4	Jumper Cables	20	\$ 1.00	https://makerstore.ca/shop/ols/products/jumper-cables-pack-of-10/v/C004-20-FF
5	HC-SR04 Ultrasonic Sensor	1	\$ 2.00	https://makerstore.ca/shop/ols/products/hc-sr04-ultrasonic-sensor
6	External structure + Bicycle holder	1	\$ 20.00	Estimated. Not included in the final product but needed for testing. Most bikes may already have a flat plate-like attachment at the back, or a basket in the front.
	Total (Before Taxes)		\$ 146.54	
	Total		\$ 164.12	<i>An estimate with +/-10% tolerance.</i>

Table 4: Bill of Materials

6.3 Testing & Validation

The plan for verification of the solution, mirroring the implementation strategy, was divided into six stages. Every verification stage is essentially a (series of) test(s) of the solution stage. The table below captures the proposed verification and validation (V&V) strategy.

Note: Each step of the V&V plan may have sub-tests that have been abstracted away from the table. Similarly, FSR (Functional System Requirement) and NFSR (Non-Functional System Requirement) are the requirements derived by the team from the user needs which are not included in this document.

Implementation Stage	Verification Plan	Validation Plan
<u>Stage 1</u> Android mobile app and sensor data.	<u>Stage 1</u> 1. Code syntax and logic checks.	Test against NFSR01, FSR01 and FSR02.

	<ol style="list-style-type: none"> 2. Check results on both emulator and a physical device. 3. (Re)define metrics, baselines or thresholds and check obtained results against them. 4. Check if values are being stored either locally or to a server, as the team intended. 	
<u>Stage 2</u> Data modelling.	<u>Stage 2</u> <ol style="list-style-type: none"> 1. Check the resolution of the modelled information. 2. Check if anomalies register clearly in the data. 	Test against FSR01.
<u>Stage 3</u> Identification and sourcing of sensors and microcontroller(s).	<u>Stage 3</u> <ol style="list-style-type: none"> 1. Check if identified sensors match the proposed system architecture. 2. Check if the required equipment is within the proposed/allocated budget. 	Test against FSR03.

	3. Check if the acquired sensors are of good quality.	
<u>Stage 4</u> Code for sensors and microcontroller.	<u>Stage 4</u> 1. <i>Essentially the same as the steps for stage 1.</i> 2. Test all sensors at the same time (real-time test). 3. Explore the possibility of uploading values to a global location that can ideally be accessed by the mobile app.	Test against FSR03.
<u>Stage 5</u> Interfacing the microcontroller with the mobile app.	<u>Stage 5</u> 1. Check if the mobile app can turn on and off the enable signals of the microcontroller – logical.	Test against FSR03 and FSR05.
<u>Stage 6</u> Housing for the equipment. Housing the sensors and microcontroller with an independent power source.	Stage 6 1. Check size constraints. 2. Check environmental factors.	Test against FSR04.

Table 5: Verification and Validation Strategy

6.3.1 Performed V&V

Staying true to both the identified design principle of Keep It Simple, Silly (KISS) and the adopted (model-based) systems engineering methodology of ARCADIA, most of the verification was performed in the system architecture of the system model in Capella. The requirements were captured in the system model and tagged to solution architecture elements, providing clear traceability and verification.

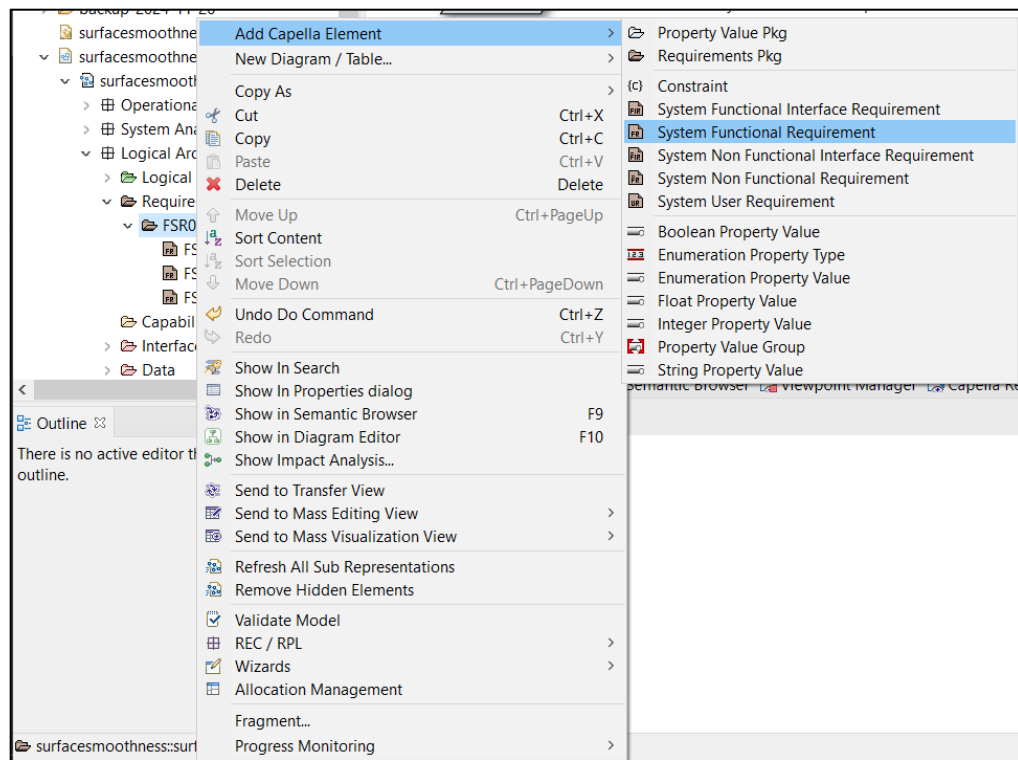


Figure 12: Example of adding functional and non-functional requirements to the system model

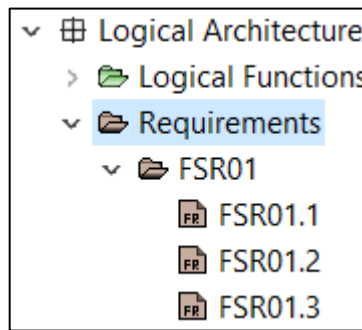


Figure 13: Example of requirement captured in the model

	FSR01.1	FSR01.2	FSR01.3
Record 3D motional data			
Send 3D mot. data			
Record 3D motional data		X	
Send 3D mot. data			
Record 3D positional data		X	
Send 3D pos. data			
Data reception			
Processing			
Data transmission			
Capture distance data	X		

Figure 14: Verification performed in the architecture

The software code was also split into various functions; the Android code in written in Java was coded using functions mirroring those captured in the system model. Every green box in the system architecture view is a system function. At the lower levels, these system functions can be broken down into domain-specific functions – for example, electrical, software, hydraulic and so on. This breakdown has been abstracted out for this report, but the verification was performed using the system level functions in the matrix style discussed above.

The verification matrix generated in this manner helped the team verify all items. The table below indicates successful or unsuccessful verification of the item.

Note: The items in bold failed the verification test or are unverified at the time of submission of this report. More information about these items is included at the end of the table.

Implementation Stage	Verification	Validation
<u>Stage 1</u> Android mobile app and sensor data.	<u>Stage 1</u> 1. Code syntax and logic checks. 2. Check results on both emulator and a physical device. 3. (Re)define metrics, baselines or thresholds and check obtained results against them. ^[#1] 4. Check if values are being stored either locally or to a server, as the team intended.	Test against NFSR01, FSR01 and FSR02.
<u>Stage 2</u> Data modelling.	<u>Stage 2</u> 1. Check the resolution of the modelled information. 2. Check if anomalies register clearly in the data.	Test against FSR01.
<u>Stage 3</u>	<u>Stage 3</u>	Test against FSR03.

Identification and sourcing of sensors and microcontroller(s).	<ol style="list-style-type: none"> 1. Check if identified sensors match the proposed system architecture. 2. Check if the required equipment is within the proposed/allocated budget. [#2] 3. Check if the acquired sensors are of good quality. 	
<u>Stage 4</u> Code for sensors and microcontroller.	<u>Stage 4</u> <ol style="list-style-type: none"> 1. <i>Essentially the same as the steps for stage 1.</i> 2. Test all sensors at the same time (real-time test). 3. Explore the possibility of uploading values to a global location that can ideally be accessed by the mobile app. 	Test against FSR03.
<u>Stage 5</u> Interfacing the microcontroller with the mobile app.	<u>Stage 5</u> <ol style="list-style-type: none"> 1. Check if the mobile app can control the enable 	Test against FSR03 and FSR05.

	signals of the microcontroller. [#3]	
<u>Stage 6</u> Housing the sensors and microcontroller with an independent power source.	<u>Stage 6</u> 1. Check size constraints. [#4] 2. Check environmental factors. [#5]	Test against FSR04.
<u>Remarks:</u> [#1]: Since all tests were conducted under laboratory conditions, this criterion could not be verified in the real-world. [#2]: Since not all stages were implemented, this criterion was not considered. [#3]: Since stage 5 was not implemented, this check was skipped. [#4]: Since stage 6 was not implemented, this check was skipped. [#5]: Since stage 6 was not implemented, this check was skipped.		

Table 6: Verification results

Under laboratory conditions, both indoors and outdoors, the validation of only stage 1, stage 2 and stage 3 was performed. The validation test follows the plan outlined in the table above, and involves alpha testing – testing by members of the team. Client validation has not been performed at the time of the submission of this report.

7 Conclusions and Recommendations for Future Work

7.1 Lessons Learned

The team learned the following lessons worth mentioning while working on the project.

- Fix the problem statement (no ambiguity) before starting work on a solution or concept.
- There will always be something better than the current idea, product or concept but chasing after those “optimum” alternatives leads one down a rabbit hole.
- Typically, the best advice to follow at any stage is to keep things as simple as possible unless the business or use case demands otherwise; “reinvention of the wheel” is not necessary.
- Dedicate enough amount of time for research into existing literature or work performed in the area.

7.2 Planned Work

The following system capabilities have been successfully implemented at the time of submission of the project report.

- Continuously record and monitor acceleration values using a mobile phone application.
- Continuously – or at time intervals required by the use case – record GPS coordinates using the mobile phone application.
- Identify anomalies using the acceleration values.
- Drop GPS markers on the UI at regular intervals or at detection of anomalies.
- Continuously record distance measurement using the ultrasound sensor connected to the Raspberry Pi.
- Record image on-demand using the camera sensor and Raspberry Pi.

At the time of submission of the prototype report, the following activities are pending. Due to various reasons stemming from the constraint of lack of time and resources, the following activities

could not be performed in the duration of the course GNG 5140 for the fall 2024 semester. Future work on this project, with the help of the information and resources included on MakerRepo, may resume from the following activities.

- Make the Raspberry Pi “headless” – Remove dependency on the need for an external display, mouse and keyboard.
 - Identified strategy – Include the project logic into a service that is called on boot.
- Detailed real-world testing – Verification and validation activities, plus user testing
- Implementation of stage 5 – Building an interface between the mobile application and Raspberry Pi
 - Identified strategy – Use Google Firebase (same used for backend) and perform data integration on the server or on client-side. This is an asynchronous activity and should not affect the real-time collection of data on the side of the client.
- Implementation of stage 6 – Weatherproof housing for the sensors and microcontroller including dedicated power.
- Political considerations – Bikeways in Ontario may be removed (at the time of writing this report).
- Performing continuous improvement using the Kaizen cycle.

8 Bibliography

This bibliography is taken as-is from the project report document. The number indicated against each entry is of no significance to the information contained within this user manual.

[1] Löw, Pablo & Krisp, Jukka. (2024). Smoothing the Ride: A Surface Roughness-Centric Approach to Bicycle Routing. AGILE: GIScience Series. 5. 1-6. 10.5194/agile-giss-5-39-2024.

[2] Anna Niska, Leif Sjögren, Peter Andrén, Christian Weber, Tineke de Jong, Aslak Fyhri, Determination of riding comfort on cycleways using a smartphone application, Journal of Traffic and Transportation Engineering (English Edition), Volume 11, Issue 4, 2024, Pages 747-760, ISSN 2095-7564, <https://doi.org/10.1016/j.jtte.2023.05.010>.

[3] Toljic, Marko & Brezina, Tadej & Emberger, Günter. (2019). The influence of surface roughness on cyclists' velocity choices. Municipal Engineer. 174. 2-13. 10.1680/jmuen.18.00058.

[4] Zang K, Shen J, Huang H, Wan M, Shi J. Assessing and Mapping of Road Surface Roughness based on GPS and Accelerometer Sensors on Bicycle-Mounted Smartphones. *Sensors*. 2018; 18(3):914. <https://doi.org/10.3390/s18030914>.

[5] Shields M, Connor Gorber S, Janssen I, Tremblay MS (September 2011). "Bias in self-reported estimates of obesity in Canadian health surveys: an update on correction equations for adults" (PDF). Health Reports. **22** (3): 35–45.

[6] EXA Tools (2016) *Vibration meter - apps on Google Play*, Google. Available at: <https://play.google.com/store/apps/details?id=com.exatools.vibrometer> .

[7] Ontario Ministry of Transportation. (2013). *Ontario Traffic Manual, Book 18: Cycling Facilities*.

- [8] Random Nerd Tutorials (2013) Complete Guide for Ultrasonic Sensor HC-SR04. Available at: <https://randomnerdtutorials.com/complete-guide-for-ultrasonic-sensor-hc-sr04/> .
- [9] Dewalt. (n.d.) 100 ft. Laser Distance Measurer. Available at: <https://www.dewalt.com/product/dwht77100/100-ft-laser-distance-measurer>.
- [10] Bosch. (n.d.) Blaze 100ft Red Beam Laser Measure. Available at: <https://www.northerntool.com/products/bosch-blaze-100ft-red-beam-laser-measure-max-measuring-distance-100-ft-accuracy-0-0625-in-model-glm100-23-5776799#hotbar-description>.
- [11] Miles Sey. (n.d.) Outdoor Laser Measure Camera P7AK. Available at: <https://mileseeytools.com/products/outdoor-laser-measure-camera-p7ak>.
- [12] Charcity. (n.d.) Distance Measuring Tool. Available at: <https://www.amazon.ca/Charcity-Distance-Waterproof-Pythagorean-Measuring/dp/B0CF25L9V8>.
- [13] Merritt DK, Chang GK, Rutledge JL. Best practices for achieving and measuring pavement smoothness, a synthesis of State-of-Practice (2015). Louisiana Transportation Research Center. FHWA/LA.14/550. LTRC Number: 14-1PF. State Project Number: 30001420. https://rosap.nrl.bts.gov/view/dot/28837/dot_28837_DS1.pdf .
- [14] Statcounter, a web analytics service that uses website trackers to obtain statistics, <https://gs.statcounter.com/os-market-share/mobile/worldwide> .

APPENDIX

This project is hosted on MakerRepo, which is essentially a project hosting and collaboration platform managed by the Center of Engineering and Entrepreneurship Design at the University of Ottawa. The repository for this project contains all resources including design files and reports. As a result, only the link to this repository is included below.

Document Name	Document Location and/or URL	Issuance Date
MakerRepo	https://makerepo.com/bsaditya/2362.gng5140-cycling-facility-surface-smoothness-measurement	27 November 2024

Table 7: Reference Resources