Software Requirements Specification

for

SMART ARDUINO HELMET

Version 1.0

Prepared by

Group Name: Group 19

AYEBARE ATUHAIRE EUNICE	24/U/04071/PS	ayebeunatu@gmail.com
TUSIIME MARK	24/U/11684/PS	tusiimejohnmark@gmail.com
OKURE ENOCK	24/U/10690/PS	okureenock@gmail.com
WEREDWONG J.PRECIOUS	24/U/20747/PS	jjpreciousweredwong@gmail.com
NAMUYIMBWA MARTHA	24/U/09436/PS	kukirizamartha@gmail.com

Mentor: *Mr. Denish Azamuke*

Course: CSC 1304 Practical Skills Development

Date: 28/07/2025

Contents

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
1.0	**Group 19**	Initial completed version for submission	**26/07/2025**

1 Introduction

The Smart Arduino Helmet is an innovative safety solution designed to address the high incidence of boda accidents in Uganda, especially in densely populated areas. Many accident victims are unable to call for help, and emergency responses are often delayed. This project proposes a smart helmet using an Arduino based system to detect accidents, alert emergency contacts via a GSM, and help save lives through immediate action. This section introduces the purpose of this SRS document, the scope of the Smart Arduino helmet system, its intended audience, and the terminology used throughout. The goal is to guide both technical and non-technical stakeholders in understanding the objectives and requirements of the system.

1.1 Document Purpose

This Software Requirements Specification document describes the functionalities, interfaces, and requirements of the Smart Arduino Helmet system. The helmet is a student-developed prototype aimed at detecting accidents and alerting emergency contacts automatically through a GSM module.

The purpose of this document is to provide a clear, comprehensive description of the product's intended behaviour and environment. It serves as a communication tool between the development team and the client (course facilitator), and as a blueprint for the system's design, development, testing, and maintenance.

1.2 Product Scope

The Smart Arduino Helmet is an embedded system that enhances rider safety by detecting crashes and inactivity, then automatically sending a location-based emergency SMS to a predefined contact. It also includes a manual panic button for the rider to request help or cancel a false alarm. The system relies on a combination of Arduino, accelerator, GSM module, and GPS module, all integrated within the helmet without altering its physical structure.

Key benefits include:

- Faster emergency response through automatic alerts
- Greater safety for boda riders
- Low-cost solution that can be deployed widely
- Potential to reduce fatalities and serious injuries

In addition to the embedded smart helmet system, the project includes a companion Android mobile app that allows users to interact with the helmet. The App provides a user-friendly interface to

- Pair with the helmet via Bluetooth
- Receive crash notifications
- Configure emergency contact numbers
- Monitor helmet system status

1.3 Intended Audience and Document Overview

The primary audiences for this document are:

- Course Facilitators: To assess the team's understanding of software engineering principles.
- Developers: To use as a guide throughout the software design and implementation
- Testers: To refer to during validation and verification
- Potential Stakeholders/Clients: NGOs or road safety organisations may also use this document to understand the system for possible funding or deployment

The document is organized to introduce the system (Section 1), provide an overview of its general features (Section 2), list detailed requirements (Section 3), and describe non-functional and other requirements (Section 4 and 5).Diagrams and use cases are included to enhance clarity.

1.4 Definitions, Acronyms and Abbreviations

GSM : Global System for Mobile Communications
GPS : Global Positioning System
SMS : Short Message Service
Boda Boda : Commercial motorcycle commonly used in East Africa
MPU-6050 : Motion tracking sensor (accelerometer + gyroscope)
Arduino : Open-Source micro controller platform
SRS : Software Requirements Specification
BLE : Bluetooth Low Energy

1.5 Document Conventions

Font: Arial, size 11
Spacing: Single-spaced with 1" margins
Headings: Follow IEEE standard section numbering (eg1,1.1,1.2)

1.6 References and Acknowledgments

- *IEEE Citation Style Guide: https://www.ieee.org/documents/ieeecitationref.pdf*
- *"Smart Arduino Helmet" internal project concept document*
- *Uganda Police Annual Traffic Reports (for accident statistics)*

2 Overall Description

2.1 Product Perspective

The Smart Arduino Helmet is a self-contained, standalone embedded system. It is not a follow-up to an existing product but is instead a novel, Arduino based prototype designed from scratch. It integrates a crash detection algorithm, GSM communication for emergency alerts, and phone GPS for location tracking, The system fits inside a motorcycle helmet without compromising the helmet's structural integrity.

The Smart Arduino helmet is composed to two main components:

1. An embedded safety system within the helmet (Arduino + sensors)
2. An Android mobile application that serves as a user interface and configuration tool

2.2 Product Functionality

The Smart Arduino Helmet includes the following key functions:

- Crash Detection: Uses accelerator data to detect a sudden impact or fall
- Inactivity Timer: Starts a countdown if no movement is detected after a crash (default:10 seconds)
- Emergency SMS: Sends the location via GSM(SIM800L) to present emergency contact
- Manual Panic Button:
    1. Single Press: Send SOS alert immediately
    2. Double Press; Cancel the alert if it was a false trigger.
- Embedded and Safe Installation: All electronics are embedded within the padding layer-no structural cutting of the helmet

New App-Specific Functions:

- Contact Configuration: Users can set or change their helmet contact through the app
- Crash alerts (App notifications): Receives in-app notification in addition to SMS
- Crash Log Viewer: Displays a history of all detected crashes
- Status Dashboard:

2.3 Users and Characteristics

User Types:

- *App User: Boda riders or their family members*
- *Secondary Users: Emergency responders(receiving SMS), potential fleet operators or NGOs*

User Characteristics:

- *Require basic smartphone familiarity*
- *Usage Frequency: Daily use, whenever helmet is worn*
- *Critical Needs: Simple interaction (e.g double press for SOS), reliable system behaviour.*

Key Design Implication:

- *The system must work autonomously, with minimal user interaction required during emergencies*

2.4 Operating Environment

In this part, make sure to include a simple diagram that shows the major components of the overall system, subsystem interconnections, and external interface

- *Hardware Platform: Arduino Uno*
- *Power Supply: 5V power source via rechargeable battery*
- *Operating Systems: Windows for programming the Arduino*
- *Communications: GSM network (SIM800L) and GPS satellite for location data*
- *Weather Exposure: The helmet is worn outdoors, so all components must be securely enclosed within padding to avoid moisture/dust exposure*
- *Mobile App:*
    - *OS: Android 8.0 and above*
    - *Required permissions: location, SMS, notifications*

Minimum Platform Requirements:

- *GSM-enabled SIM card*
- *GPS signal availability*
- *Basic mobile signal(2G)*

2.5 Design and Implementation Constraints

- Form factor Constraint: All components must fit in the helmet padding without compromising safety
- Budget Limitation: Must be built using affordable, student-accessible components
- Power Constraint: Battery-powered operation with limited runtime; energy-efficient programming is critical
- Network Dependency: GSM module depends on mobile network availability for SMS
- Real-Time Responsiveness: System must respond in real-time to accidents and inactivity

2.6 User Documentation

The following user documentation will be developed:

- *Quick Start Guide: How to wear, charge, and use helmet*
- *User Manual: Explaining panic button usage, how the system detects crashes, and how to reset*
- *Emergency Setup Manual: Instructions for entering emergency contact number and inserting the SIM card.*

These documents will be concise, visual, and translated into local languages if needed for wider adoption.

2.7 Assumptions and Dependencies

- Riders will consistently wear the helmet while riding
- There will be sufficient GSM network coverage in most operating areas
- Emergency contact numbers are correctly stored and accessible
- GPS signal will be available in urban and semi-urban areas.

- Helmet construction allows secure embedded of components without damage during use or impact

3 Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces-pic of UI

The Smart Arduino Helmet will have a minimalist interface design due to its safety-critical context. User Interfaces are limited to a single panic button and buzzer for audio feedback.

Interface Components:

- Panic Button:
  - Double Press: Sends an immediate SOS SMS
  - Triple press (within 5 seconds): Cancels the alert
- Buzzer:
  - Informs the user that the alert countdown has started
- Home screen: Status overview
- Profile screen: Input and save helmet contact number
- Notification screen: Alerts for real-time crash detection

3.1.2 Hardware Interfaces

- *Arduino Uno: Micro controller for central processing*
- *Accelerometer/ Gyroscope: Detects motion and fall*
- *SIM800L GSM Module: Send SMS to emergency contacts*
- *GPS Module: Captures location for alerts*
- *Push Button: Manual SOS and cancel trigger*
- *Battery Pack(5V): Powers entire system*

3.1.3 Software Interfaces

- *Arduino IDE (Windows): Used for writing and uploading firmware to the Arduino.*
- *Embedded C/C++ code: Software will be written using Arduino libraries*
- *GSM AT Commands: Used in code to communicate with SIM800L module*
- *Serial Communication: Used for debugging during development via USB interface*

Android APIs:

- *Shared Preferences*
- *Notification Manager for push alerts*

3.1.4 Communications Interfaces

- *GSM/SMS communication*
  - *The system includes a SIM800L GSM module for sending emergency alerts via SMS*
  - *SMS is sent automatically if:*
    - *A crash is detected and rider is inactive for 10 seconds*
    - *Panic button is pressed twice*
- *SMS content includes:*

- *Alert message*
- *GPS location link*

- *Future communication options*
- *Bluetooth Low Energy (BLE)Communication*
    - *The helmet would use a BLE module to communicate with the Android mobile App*
    - *The app scans for nearby BLE devices, connects to the helmet securely, and listens for crash status, battery level, and configuration requests.*
    - *Data exchanged is lightweight and formatted in JSON strings*
    - *Communication is one-way or bi-directional, depending on the use case:*
        - *Helmet>App: crash alerts, battery status*
        - *App>helmet: Update emergency contact, reset system*

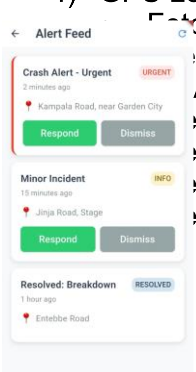
Key considerations:

- *BLE connection requires manual pairing via the app*
- *The connection is encrypted via standard Android BLE security mechanisms*
- *If the app is not connected, the helmet will still function independently*

3.2 Functional Requirements

The system will be divided into functional modules:

1) Crash Detection Module
    - Detects sudden change in acceleration
    - Starts a 10 second inactivity timer after a detected fall
    - If no movement resumes, triggers emergency alert
2) Emergency Alert Module
    - Composes an SMS message containing:
        - ✓ Alert type
        - ✓ GPS Location coordinates
    - Sends message to predefined emergency contact
3) Panic Button Handler
    - Double press sends immediate emergency SMS
    - Triple press cancels an alert within a 5 second window
    - Debounce logic included to avoid false triggers
4) GPS Location Module
    - Fetches real- time location coordinates using the phone’s inbuilt GPS



    - egrates with GSM module to embed location in SMS

App functions

- app shall allow the user to input, edit, and save an emergency contact
- app shall receive crash event data from the helmet in real-time
- app shall display crash logs with timestamps and location data
- app shall notify the user of crash detection via in-app alerts

4 Other Non-functional Requirements

4.1 Performance Requirements

- Crash response time: The system shall detect a crash and send an emergency SMS within 15 seconds of inactivity
- Location accuracy: The user's phone shall provide location data with an accuracy margin of +_10 meters under open sky conditions
- Panic button responsiveness: The system shall recognize double and triple press inputs within 2 seconds to avoid false triggering
- Startup time: The helmet system shall boot and become fully operational within 5 seconds of powering on
- Battery Life: The system shall support a minimum 8 hours of continuous standby time on a single charge
- The app UI shall render dashboard and logs in under 5 seconds

4.2 Safety and Security Requirements

Safety requirements

- *Non-invasive installation: The system shall not alter or weaken the helmets structural integrity*
- *Fail-safe SMS: In case of GSM network failure, the system shall attempt resending the emergency SMS every 30 seconds up to 3 times*
- *Manual Override: The rider shall be able to cancel false alarms with a double press of the panic button*

Security Requirements

- *User data protection*
  - ✓ *No user data will be shared externally*
- *SMS Access Control*
  - ✓ *Only authorized numbers can receive emergency messages*
- *Future enhancements*
  - ✓ *Message encryption and GPS data obfuscation*
- *Emergency contact and crash logs will be encrypted locally using AES*

4.3 Software Quality Attributes

- *Reliability*
- *The system shall operate consistently in various lighting and weather conditions*
- *Crash detection shall trigger with 95% accuracy during testing*
- *GPS and GSM modules shall recover automatically if signal is temporarily lost*
- *Maintainability*
- *The Arduino codebase will be modularised, using functions and libraries to separate crash logic, SMS handling, and GPS data retrieval.*
- *System firmware updates can be applied through USB via the Arduino IDE*
- *Portability*
- *The system can be transferred to different helmet models with minimal adjustments, provided sufficient padding space is available.*
- *Future versions can adapt to other microcontrollers like ESP32*
- *Robustness*

- *All components will be shock-mounted inside the padding to survive typical falls and vibrations*
- *The system shall restart safely if powered off accidentally*

➢ *Usability (Mobile App)*

- *App UI will follow Material design principles*
- *All major actions are accessible within 2 taps*
- *Visual cues (icons, colors) used for critical information (e.g. red for crash alert)*

➢ *Portability*

- *The app shall be compatible with any Android device running Android 8.0 or higher*
- *Future support for IOS may be considered*

5 Other Requirements

This section outlines additional considerations beyond the technical specifications already described.

5.1 Legal and Regulatory Compliance The system must comply with Uganda's national traffic safety regulations and telecom standards, especially regarding the use of GSM modules and electronic safety gear.

5.2 Environmental Considerations The helmet must be resilient in various weather conditions. Components should be protected against dust and humidity using waterproof casing or sealed padding.

5.3 Internationalization Although the primary deployment is in Uganda, future versions of the system may require multilingual support in both the mobile app and user documentation to cater to riders in other East African countries.

5.4 Ethical and Social Impact The system must respect user privacy. No data should be collected or transmitted without explicit user consent. Emergency messages should only be sent in critical situations.

5.5 Data Handling and Logging All crash events are logged locally within the app. In future, logs may be synchronized to cloud services such as Firebase for analytics and safety audits.

5.6 Scalability and Reuse The Smart Arduino Helmet system can be scaled to support commercial fleets or integrated into NGO safety initiatives. The architecture allows for hardware upgrades (e.g., ESP32) or the addition of new features (e.g., helmet lock).

Appendix B - Group Log

- 15/06/2025: Initial brainstorming – Team met to identify major problems boda riders face; discussed real-world accident scenarios and safety gaps.
- 22/06/2025: Component research – Selected MPU-6050, SIM800L, and Arduino Uno; evaluated pricing and availability.
- 01/07/2025: System design – Agreed on the interaction flow between sensors, GSM, and mobile app.
- 10/07/2025: Development – Built and tested initial prototype; encountered and resolved GPS latency issues.
- 14/07/2025: App design – Created BLE connectivity logic; defined UI wireframes and crash alert flows.
- 20/07/2025: Testing phase – Simulated falls and verified alert accuracy; tested panic button timing.
- 24/07/2025: User documentation – Drafted quick start and emergency setup guides.
- 27/07/2025: Final review – Verified all requirements, updated documentation, and prepared SRS for submission.