Software Requirements Specification

For

Elevate

Instructor: Professor Bruce Char
Team Members: Jake Rauchen, Kevin Tayah, Matthew Horger
Cycle: 1
Date Submitted: ~~1/29/2017~~ 4/5/2017

Grading Rubric - Requirements Specification

This rubric outlines the grading criteria for this document. Note that the criteria represent a plan for grading. Change is possible, especially given the dynamic nature of this course. Any change will be applied consistently for the entire class.

Achievement	**Minimal**	**Exemplary**	**Pts**	**Score**
Content (80)	Section(s) missing, not useful, inconsistent, or wrong.	Provides all relevant information correctly and with appropriate detail		
Introduction Scope Definitions			10	
User Profile			20	
Functional Requirements			30	
Performance & Design Requirements			10	
Data Requirements			10	
Writing (20)				
Grammar and Spelling	Many serious mistakes in grammar or spelling	Grammar, punctuation, and spelling all correct	10	
Expression Tone	Hard to follow or poor word choices Tone not appropriate for technical writing	Clear and concise. A pleasure to read Tone is consistently professional	5	
Organization Layout	Information difficult to locate Layout is inconsistent, visually distracting, or hinders use	All information is easy to find and important points stand out Layout is attractive, consistent, and helps guide the reader	5	
Late Submission			-10 -25	
Total			100	

1 Introduction

This section provides an overview of Elevate. Scope provides a short description of the product and how it is useful; Definitions explains terms with which a reader may not be familiar; and User Profile identifies the ways different groups of people would make use of Elevate.

1.1 Scope

This project will develop a modular helmet that allows users to detect blind spots, record collisions, and monitor GPS coordinates and features to track their rides. Users, both professional and recreational, will be able to improve their safety and pleasure in terms of their riding experience with elevate

1.2 Definitions, Acronyms, and Abbreviations

In this project, we will be using various pieces of technological hardware modules that will achieve our goals. An Arduino ~~Nano~~Uno, a micro-processor, will be the heart of our helmet and will be referred to as the Arduino throughout this proposal. A HC-SR04 module detects objects in its' range of sensors, which will be used for our blind spot detection. This module will be referred to as our ultrasonic modules. Plone, a content management system, will be used to store our data from the helmet and application. This module will be referred to as Plone CMS. Finally, our mobile application will be accessing Google's Maps JSON data to track GPS coordinates. For the sake of this document, this data will be referred to as the Maps API.

1.3 User Profile

Bicyclist – A professional bicyclist will be able to use Elevate to safely monitor his or her surroundings while riding and can navigate more efficiently in their route by not having to look away from the road when checking for blind spots. Also, the increasing number of those who commute on bicycles can use Elevate to increase their safety.

Skater – A recreational or professional skater will be able to use Elevate for the same purposes as a bicyclist. However, there will also be an option to view areas that have increased elevation to find enjoyable hills.

Training Athlete – An athlete can use Elevate as a bicyclist and skater do, but also will be able to better customize their route which will provide them with a more desirable workout that is created for them personally.

Retail Chain – A commercial store can benefit from Elevate by customizing and designing functional helmets for sale to riders across neighborhoods.

Professional Sports Organization – A competitive organization can use Elevate to boost safety of participants and track better analytics in terms of performance and times for participants to reach certain checkpoints using GPS tracking.

2 External Interfaces

This section identifies ways in which Elevate interacts with people and other systems.

2.1 User Interface

Elevate will interact with users two ways. One will be via the modular helmet. Upon tracking of blind spots, the helmet will noticeably beep for the rider when an object has been detected. The helmet will also beep when collision has been detected via a contact switch.

Another way will be through the mobile application. This application will have several abilities, such as elevation tracking, route mapping, and more~~,~~ that the user can interact with. The application will be visually designed to be user-friendly and simple to use, while maintaining functionality.

2.2 Data Interface

Elevate will obtain elevation readings and GPS coordinates from Google Maps API. The helmet will also obtain distance and measures of impact for analysis. This data will be stored in some type of content management database to allow for Google Maps overlay to match coordinate data.

3 Specific Requirements

3.1 Functional Requirements

The statements below define the functional requirements for the system.

1 – Hardware Modules

Elevate will provide seamless communication between connected modules located in the helmet. Elevate must make sure all modules are working properly and can communicate with the user upon ~~testing.~~usage.

2 – Mobile Application

Elevate will provide a way for user groups to easily view route data with details, such as elevation. Elevate must make sure that data being received from the helmet and the Maps API is displayed to the user accurately. The application will give the user the ability to set routes between locations and will allow them to choose the route that they desire, based on elevation change, distance, etc.. It will be visually/functionally designed for quick and easy use as users may be using the application while riding.

3 – Database

Elevate will provide backend support via Plone CMS for managers to view helmet data and Google Maps to view where the helmet is. This support can be used to allow for users to retrieve their helmet if they lose it. This support can also be used to allow for

competitions for tracking participants. Plone CMS must also be able to be accessed via any computer with proper account logins.

3.2 Performance Requirements

The statements below define the performance requirements for the system.

1 – Ultrasonic Modules

Distance modules in the helmet must be able to detect distances ~~5 feet~~25 meters behind the user ~~(Note: 5 feet is the baseline and will be adjusted upon testing)~~ and can consistently refresh detections within 1 millisecond ~~.~~ The module must be able to alert the user within 1 second or less.

2 – GPS Module

GPS data being collected from the helmet must be accurate within 10 feet ~~(Note: 10 feet is the baseline for accurate map readouts and can be refined later)~~. GPS data sent to Plone CMS ~~the mobile application~~ must be sent within 30 seconds or less for seamless user experience and tracking efficiency.

3 – Collision Module

When the helmet is collided with an object, the sensor must be able to timestamp and record the amount of impact that occurred. The sensor must also be able to accurately send an alert to the user saying that the data was successfully saved.

4~~3~~ - Helmet

The helmet must be designed with the modules so that retail locations can design and sell functional helmets.The helmet must also have some sort of weatherproof feature so that the modules do not get damaged by moisture.

5 - Database

Plone CMS must have created accounts for users or developers to login into the system. Plone must also be designed such that any computer can access the database with the proper account credentials. Plone must also be able to overlay Maps API data and parse data real-time for waypoint tracking.

6~~4~~ - Mobile Application

The mobile application must be able to store and access Maps API data within a timeframe of a minute for users to enjoyably use the application without significantly long waiting times. The application must be designed so that users have no trouble using its' features. Also, it must be very responsive because users may be using the application while they are riding and should not be distracted.

7 – Batteries

The helmet must have rechargeable batteries that have a usage span of over 8 hours. If power decreases, the Arduino and ultrasonic sensors must still work; collision and GPS will be ceased. If users choose to use disposable batteries, the Arduino must regulate

voltage appropriately.

3.3 Design Constraints

3.3.1 Constraint: Elevate will be an android app.

Reason: Functional usage with Google Maps API and application design will be better with Android.

3.4 Data Requirements

1 - Helmet

Name	Type	Size	Comment
Ultrasonic Module	Number		Provides a distance reading so that our alarm can alert the user when that number reaches a certain point
Collision Module	Number		Provides users with a number of the amount of force they endured upon impact
GPS Breakout	JSON		Provides mobile application with GPS Coordinates and analytic tracking

2 – Mobile Application

Name	Type	Size	Comment
Elevation Readings	Button / JSON		Upon clicking, will display appropriate data from helmet / Maps API with user requests
Route Mapping	Button / JSON		Upon clicking, will process user request and map desired route with given details
Application Development	Java		Small tasks from user interaction with the application will be executed using code written in Java

3 - Database

Name	Type	Size	Comment
GPS Data	JSON		Stored in MySQL database using Zope
Map and waypoint Overlay	JSON		Upon login, will present default map with helmet waypoint
Accounts	SSH		User and developer account logins