NextGen Wheelchair: A Semi-Autonomous Approach for Disabled People Using Head Motion and Digital Twin Technology

Software Testing Document

Integrated Design Project

CSE – 460

Group: A1

Group Members:

Abrar Faiyaz Khan (202014002)

M Rayhan Ferdous Faisal (202014022)

Mohiuddin Bilwal (202014037)

Easin Arafat (202014049)

Shekh. Md. Saifur Rahman (202014051)

Table of Contents

1. Introduction

Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not. In simple words, testing is a measurement of executing a system in order to identify any gaps, errors, or missing requirements contrary to the actual requirements. According to ANSI/IEEE 1059 standard, Testing can be defined as - A process of analyzing a software item to detect the differences between existing and required conditions (that is defects/errors/bugs) and to evaluate the features of the software item. Documentation for Software testing helps in estimating the testing effort required, test coverage, requirement tracking/tracing, etc. The Software Test Plan (STP) is designed to prescribe the scope, approach, resources, and schedule of all testing activities. The plan must identify the items to be tested, the features to be tested, the types of testing to be performed, the personnel responsible for testing, the resources and schedule required to complete testing, and the risks associated with the plan.

1.1 Objective

Testing is the process of analyzing a software product to find inconsistencies between current and desired circumstances and to assess the software product's features. This system's advancement, which is based on head movements, aims to increase the mobility of paralyzed patients, particularly those who are unable to move by utilizing their own muscles. Additionally, our technique strives to lessen the physical strain placed on the caregivers who assist disabled individuals in using their wheelchairs. Any emergency case, such as a patient being unconscious or a system suddenly crashing while in motion, will cause the system to stop immediately. The system can be controlled and monitored via a web interface and two alternative modules – voice speech and joystick are also used to control the system. Moreover, the system will be stopped whenever it'll detect any obstacle while moving. A web interface is available for controlling and monitoring the system, and two additional control modules – voice speech and joystick – can also be employed. Moreover, whenever the system detects an obstruction while moving, it will stop.

1.2 Scope

The overall purpose of testing is to ensure the application meets all of its technical, functional, and business requirements. The purpose of this document is to describe the overall test plan and strategy for testing the application. The approach described in this document provides the framework for all testing related to this application.

2. Test Items

We will mainly focus to ensure the following items are tested.

- Requirements specification
- Availability
- Response time
- Security
- Capacity
- Usability

• Safety
• Verification and Validation plans.

3. Features to be tested

The features to be tested are listed below:

3.1 Login System

Criteria Assessment: Usability, Security, Verification, and Validation Plans
Testing Type: Manual Testing and Usability Testing

3.2 Head Motion Detection

Criteria Assessment: Requirements Specification, Response Time, Usability
Testing Type: Unit Testing and Integration Testing

3.3 Voice Speech Recognition

Criteria Assessment: Requirements Specification, Response Time, Usability
Testing Type: Unit Testing and Integration Testing

3.4 Manual Control

Criteria Assessment: Requirements Specification, Usability
Testing Type: Unit Testing and Integration Testing

3.5 Obstacle Detection

Criteria Assessment: Requirements Specification, Safety, Usability
Testing Type: Unit Testing and Integration Testing

3.6 Emergency Halt

Criteria Assessment: Requirements Specification, Safety, Usability
Testing Type: Unit Testing and Integration Testing

3.7 Battery Percentage

Criteria Assessment: Requirements Specification, Usability, Availability
Testing Type: Unit Testing and Integration Testing

3.8 Raspberry Pi Integration

Criteria Assessment: Requirements Specification, Usability, Verification and Validation Plans
Testing Type: Integration Testing

3.9 Microcontroller Integration
Criteria Assessment: Requirements Specification, Usability, Verification, and Validation Plans
Testing Type: Integration Testing

3.10 Wi-Fi Connection
Criteria Assessment: Availability, Response Time, Usability, Security
Testing Type: Unit Testing and Integration Testing

4. Software Testing

4.1 Development Testing
Tested we've done during the development phase are discussed below:

(1) Unit Testing: Unit Testing is a level of software testing where individual units/ components of software are tested. The purpose is to validate that each unit of the software performs as designed. We've done unit testing on the following features:

1) Head Motion Detection
2) Voice Speech Recognition
3) Manual Control
4) Obstacle Detection
5) Emergency Halt
6) Battery Percentage
7) Wi-Fi Connection

(2) Integration Testing: Integration Testing is a level of software testing where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. We have done integration testing on the following features of our software:

1) Head Motion Detection
2) Voice Speech Recognition
3) Manual Control
4) Obstacle Detection
5) Emergency Halt
6) Battery Percentage
7) Raspberry Pi Integration
8) Microcontroller Integration
9) Wi-Fi Connection

(3) Manual Testing: Manual Testing is a software testing approach where tests are executed manually by human testers without the use of automated tools. The purpose of this level of testing

is to identify defects, evaluate system behavior, and ensure the usability and quality of the software through real-user interactions. We have done manual testing in Login System feature only.

(4) Usability Testing: Usability Testing is a testing method focused on evaluating a system's user-friendliness and ease of use. The purpose of this level of testing is to assess the system's user interface, interaction flows, and overall user experience to identify usability issues and make improvements for enhanced user satisfaction. We have done usability testing in Login System feature only.

5. Hardware Testing

Without stable hardware, any program will fail. The frustration and expense of supporting bad hardware can drain an organization, delay progress, and frustrate everyone involved. So, we have done the following hardware testing for ensuring a high-performance tool:

1) Wi–Fi Connection: Wi–Fi is an important part of our system. We integrated various libraries and packages from the internet for our website. Besides, the commands for controlling the wheelchair from the website were sent to the microcontroller through a smooth Wi–Fi connection. For this, we've done integration testing multiple times to ensure a secure internet connection.

2) USB Camera, Microphone, and Joystick: We've conducted multiple tests to get video feedback of the user to the website from the USB camera in our system. Besides, we've ensured smooth voice recognition and manual control commands from a microphone and joystick module respectively.

3) Giving Power to the System: We've ensured to give enough power supply to all the motors used in the system to drive continuously. Besides, a continuous power supply is ensured to the microcontroller and Raspberry – pi module to run the system with good accuracy.

4) Obstacle Detection: We've used four sonar sensors on four sides of the system to ensure the system stops instantly whenever it detects any obstacle on the path. We've done multiple integration and unit tests to get better performance from the system.

5) Microcontroller Integration: We've integrated a microcontroller with the website to send the direction commands directly from the website. Then, we uploaded the required Arduino code to the microcontroller so that the system moves in the required direction. For this, we've done multiple integration tests.

6) Raspberry – Pi Integration: We've used Raspberry – pi to host the website on the server so that it can be run from any device within a range.

6. Pass / Fail Criteria

System Moving Through Head Motion: In Table – 1, we can see that the optimum trial to successfully move the system is 3, but the mean trial came out to be 4 since the head motion was detected later than anticipated.

System Moving Through Head Motion (Table – 1)			
Participant	Number of Testing	Trials to Success	Completion Time
1	5	4	22.56
2	5	3	15.23
3	5	5	25.11
4	5	4	20.53
5	5	4	19.37
n = 5		4 ± 0.6 (Mean $\pm$ SD)	20.56 ± 3.30 (Mean $\pm$ SD)

System Moving Through Speech Recognition: In Table – 2, we can see that the optimum trial to successfully move the system is 4, but the mean trial came out to be 4.4 since the voice was sometimes lagging in the testing phase but became better as the test phase went on.

System Moving Through Speech Recognition (Table – 2)			
Participant	Number of Trials	Trials to Success	Completion Time
1	5	4	14.25
2	5	5	17.55
3	5	4	13.23
4	5	4	22.44
5	5	5	21.08
n = 5		4.4 ± 0.24 (Mean $\pm$ SD)	17.71 ± 4.28 (Mean $\pm$ SD)

System Moving Through Manual Control: In Table – 3, we can see that the optimum trial to successfully move the system is 5, and the mean trial time came out to be 5, which is the best so far. The participants performed too well in this test scenario.

System Moving Through Manual Control (Table – 03)			
Participant	Number of Trials	Trials to Success	Completion Time
1	5	5	12.14
2	5	5	9.15
3	5	5	8.76
4	5	5	9.11
5	5	5	10.89
n = 5		5 ± 0 (Mean $\pm$ SD)	10.01 ± 1.94 (Mean $\pm$ SD)

Obstacle Detection: In Table – 4, we can see that the optimum trial to successfully move the system is 4, but the mean trial came out to be 4.6 since the obstacles were not detecting at some points.

System Stopping After Detecting Any Obstacle (Table – 04)			
Participant	Number of Trials	Trials to Success	Completion Time
1	4	4	0.14
2	4	4	0.16
3	4	4	0.22
4	4	4	0.80
5	4	4	0.80
n = 5		4 ± 0 (Mean $\pm$ SD)	0.17 ± 0.02 (Mean $\pm$ SD)

Emergency Halt: In Table – 5, we can see that the optimum trial to successfully move the system is 5 and the mean trial time came out to be 5, which is again, another best test trial case.

System Getting Halted When Any Emergency Case Arises (Table – 05)			
Participant	Number of Trials	Trials to Success	Completion Time
1	4	4	4.50
2	4	4	2.24
3	4	4	3.36
4	4	4	5
5	4	4	3.53
n = 5		4 ± 0 (Mean $\pm$ SD)	3.726 ± 0.9 (Mean $\pm$ SD)

Speed Control: In Table – 6, we can see that the optimum trial to successfully move the system is 5 and the mean trial time came out to be 5, which is again, another best test trial case.

System Getting Halted When Any Emergency Case Arises (Table – 05)			
Participant	Number of Trials	Trials to Success	Completion Time
1	4	4	2.25
2	4	4	1.56
3	4	4	3.11
4	4	4	1.56
5	4	4	3.07
n = 5		4 ± 0 (Mean ± SD)	2.25 ± 0.55 (Mean ± SD)

7. Testing Schedule

The test was done according to the following schedule:

Unit Testing	10 Weeks
Code Implementation	10 Weeks
Components testing	2 Weeks
Components Overview	2 days
Modifying Codes and Hardware	10 Weeks
Designing UI	3 Weeks
Integrating Software with Hardware	1 Week
Unit Testing Complete	0 days
Integration Testing	10 Weeks
Test all Control Modules	2 Weeks
Identify Bugs	10 Weeks
Modify Code	10 Weeks
Testing the system	10 Weeks

Fig: Testing Schedule

8. Environment Requirements

8.1 Hardware

The following items were required in the hardware part of our system:

1) ESP32 Microcontroller
2) Sonar Sensors
3) 12V Battery
4) Joystick Module
5) Microphone
6) USB Camera

7) Push Switch
8) Buck Converter LM2596
9) 4 – LED 12V Battery Indicator
10) Active Buzzer
11) Resistor
12) Raspberry – Pi

8.2 Software

The following items were required in the hardware part of our system:
1) Monitor (PC/Laptop)
2) VS Code
3) Arduino
4) Microsoft Project
5) Jira

8.3 Tools

Tools we've used are:
1) Bootstrap
2) JavaScript
3) Python
4) Git and GitHub

8.4 Risks

There're some particular precautions that needed to be considered:

Risk Description	**Mitigation Plan (What to do to avoid the risk occurring)**	**Contingency Plan (What to do if the risk occurs)**	**Impact (What the impact will be to the project if the risk occurs)**
Obstacle detection doesn't work properly.	Sonar sensors should be connected properly to the system.	The system should be halted instantly.	Emergency situations like accidents and injuries may occur. So, the impact can be higher.
Raspberry Pi can't detect voice speech properly.	A high level of the platform can be used.	The system can be controlled through the other two modules.	The impact will be less.
Internet connection may not be available for long distances.	The system can be operated on a fixed range.	The system will have to be controlled by a continuous internet connection.	The impact will be less.

Fig: Resource Overview