McMaster University

SmartServe

Software & Mechatronics Capstone

Verification and Validation

Authors:
Christopher McDonald
Harit Patel
Janak Patel
Jared Rayner
Nisarg Patel
Sam Hamel
Sharon Platkin

Professor:
Dr. Alan Wassyng

Teaching Assistants:
Bennett Mackenzie
Nicholas Annable
Stephen Wynn-Williams
Viktor Smirnov

Last compiled on February 17, 2018

Contents

1 Executive Summary of Testing **2**

2 Introduction **2**
2.1 Project Overview 2
2.2 Document Overview 2
2.3 Naming Conventions and Terminology 2

3 Testing Philosophy **3**
3.1 Approach 3
3.2 Schedule 4
3.3 Environment 4

4 Test Cases **5**
4.1 Electromechanical Subsystems 5
4.1.1 Shooting Mechanism 5
4.2 Software Subsystems 7
4.2.1 Computer Vision 7
4.2.2 ShotRecommender 12
4.2.3 Shooting Model 15
4.2.4 Data Storage 16
4.2.5 User Interface 18
4.2.6 SmartServe 20

5 Test Case-Requirement Traceability Matrix **32**

List of Figures

1 Revision History 1

Date	**Revision**	**Comments**	**Author(s)**
Feb 1, 2018	1.0	Document structure and Headings	Christopher MCDonald

Figure 1: Revision History

1 Executive Summary of Testing

The testing completed on February 17th, 2018 has yielded 44 passes and 9 failures. The failures in their entirety can be attributed to features not yet implemented including; automated roll and pitch as well as the machine learning model. Given that the failure rate is roughly 17%, the team does not recommend operation of the project at this time.

An important note to consider is that most errors are localized in the Shooting Mechanism and ShotRecommender subsystems. The team would also like to expand testing further to have higher coverage of the system and increase their confidence in its reliability.

The team will continue developing the system until March 9th where tests will be redefined and added to satisfy the team's ideals. After which, the second round of testing will commence.

2 Introduction

2.1 Project Overview

SmartServe is an autonomous table tennis training system for table tennis players with various skill levels. SmartServe aids in diagnosing and improving a player's performance over time. The system trains table tennis players by shooting table tennis balls towards the player and detects successful returns from the player. The system can further adapt to the player's weaknesses and help them overcome it through further training. Importantly, SmartServe alleviates the problems of finding and working with a coach for players, as well as coaches trying to train multiple players simultaneously. The system will be deemed a success if the table tennis players and coaches can enjoy and see some value added by using SmartServe.

The project started at the beginning of the Fall 2017 academic term and will conclude at the end of the Winter 2018 term. In addition, the core project team consists of final year Software and Mechatronics Engineering students who are enrolled in the MECHTRON 4TB6/SFWRENG 4G06 capstone project course.

2.2 Document Overview

This document will provide details of all formal testing methods and results performed on the SmartServe system. The first part of testing includes detailing how it will be performed and the details of the system on which it is run. This matters due to details which can affect the testing outcomes like operating system, lighting or performance of hardware. The schedule for the test will also be detailed alongside the major deliverables to have clear outcomes to explain to stakeholders. The testing will be preformed off of the *master* branch as it stands during the beginning of the testing phase.

The actual testing will then be detailed as test cases based on what subsystem they are testing. As needed, the communication will be tested in between the subsystems to ensure communication is working as intended. Lastly, a test case-requirement matrix will be provided which maps what test cases test which requirement. This will give the reader a simple way to check if a requirement is fully satisfied.

2.3 Naming Conventions and Terminology

The following terms and definitions will be used throughout this document:

- **ACID**: a database transaction which is atomic, consistent, isolated and durable
- **CV**: computer vision
- **FPS**: frames per second
- **FSM**: finite state machine, shows transitions between states
- **GUI**: graphical user interface
- **IPO**: input process output
- **Pitch**: rotation along the y-axis; this rotation angle primarily dictates the range of the ball from the net to the edge of the table on the user side
- **Roll**: rotation along the x-axis
- **Shooting Mechanism**: refers to the part of the system that shoots the table tennis balls towards the user side (player) Please refer to Figure **??** for visual illustration
- **System**: encompasses both the hardware and software parts of SmartServe
- **System Side**: the side of the table where the electromechanical system is placed; it is the opposite side of the User Side Please refer to Figure **??** for visual illustration
- **TCP:** transmission control protocol
- **Team**: all team members of the core capstone project, as noted in the list of Authors
- **User Side**: the side of the table where the user (player) is standing
- **Yaw**: rotation along the z-axis; this rotation angle primarily dictates the panning functionality of the shooting mechanism from the right side to the left side of the table

3 Testing Philosophy

3.1 Approach

The approach to testing will be to separate the subsystems as much as possible and test them accordingly. Every subsystem will be unit tested in JUnit or PyUnit if implemented in Java or Python respectively. The shooting mechanism will be tested manually to ensure it is functional properly. Stubbing a service or dependancy can be done to return consistent, reliable and predictable results. For instance, the SmartServe system may use a stub for the CV subsystem to return a predefined sequence of pass/fail flags.

3.2 Schedule

Table 1: Testing Schedule

Testing Schedule		
Task	**Date**	**Notes**
Complete Test Cases	February 13, 2018	N/A
Run Tests (First)	February 16, 2018	N/A
Edit Test Cases	March 9, 2018	N/A
Run Tests (Second)	March 16, 2018	N/A

3.3 Environment

The SmartServe system uses heavy computation power due to the CV and ML models, so the system which runs the test will have the following details. The system will be a 15-inch Macbook Pro (Late 2016) running macOS High Sierra (10.13). The Macbook Pro has an 2.6 GHz Intel Core i7 CPU and a Radeon Pro 450 2GB GPU. The location will be in Thode Makerspace where the CV will be adjusted as necessary for those lighting conditions.

4 Test Cases

4.1 Electromechanical Subsystems

4.1.1 Shooting Mechanism

Test ID: SMC1 **Feeding Mechanism Rotation Test** Status: PASS	
Description: The feeding mechanism rotates by given amount in degrees.	
Pass/Fail Condition: The feeding mechanism rotates by given amount of degrees within a tolerance of 2 degrees.	
Pre-Conditions: The feeding mechanism is ready/powered on. Input: Integer indicating the amount of degrees to be rotated.	
Expected Results: Feeding mechanism rotates to the required position.	Actual Results: One ball is shot at a time
Post-Conditions: N/A	

Table 2: Feeding Mechanism Rotation Test

Test ID: SMC2 **Shooting Control Test** Status: PASS	
Description: The shooting control shoots the ball using a rotating wheel at the requested power level to achieve the desired speed.	
Pass/Fail Condition: The system must reach the same distance each time for the same amount of power, within 0.1 metres of error.	
Pre-Conditions: The Shooting Control motor is powered on, and connected to the system. Input: Integer between 0-100 indicating the power level of the shot.	
Expected Results: 2 successive shots land in the same spot	Actual Results: 2 successive shots land in the same spot
Post-Conditions: N/A	

Table 3: Shooting Control Test

Test ID: SMC3	**Four Position Roll Test**	Status: FAIL
Description: The system rotates the Shooting Control to one of the four default positions.		
Pass/Fail Condition: Rotates to the indicated position.		
Pre-Conditions: The Shooting Control motor is powered on, and connected to the system. Input: Integer between 0-3 indicating the desired default position.		
Expected Results: The shooting control rotates to the requested default position.	Actual Results: Not implemented	
Post-Conditions: N/A		

Table 4: Four Position Roll Test

Test ID: SMC4	**Adjustable Pitch Control Test**	Status: FAIL
Description: The pitch control angles the shooting mechanism at the desired pitch level.		
Pass/Fail Condition: Rotate to the desired pitch level with a tolerance of 5 degrees.		
Pre-Conditions: N/A Input: Integer between -15 deg to 45 deg indicating pitch angle.		
Expected Results: Rotate to the desired pitch level with a tolerance of 5 degrees.	Actual Results: Not implemented	
Post-Conditions: N/A		

Table 5: Adjustable Pitch Control Test

Test ID: SMC5 **Panning Shooting Mechanism Across the Table** Status: PASS	
Description: Move the system to face the direction specified in degrees.	
Pass/Fail Condition: System moved to desired position.	
Pre-Conditions: Panning stage is homed correctly before moving to the first location. Input: Integer between 60 deg to 120 deg indicating where to point the shooter.	
Expected Results: Moves to the desired position without going out of bounds.	Actual Results: Moves to desired positions
Post-Conditions: When powered down, the system rotates to the home (0 degrees) position.	

Table 6: Panning Shooting Mechanism Test

4.2 Software Subsystems

4.2.1 Computer Vision

Test ID: CV1 **Ball Detection Test** Status: PASS	
Description: Tests if CV subsystem can detect a table tennis ball	
Pass/Fail Condition: N/A	
Pre-Conditions: CV subsystem successfully connects to camera Input: Ball is placed in frame	
Expected Results: CV subsystem detects ball	Actual Results: As expected
Post-Conditions: N/A	

Table 7: Ball Detection Test

Test ID: CV2	**Ball Upward Detection Test**	Status: PASS
Description: Tests if CV Subsystem can detect upward motion of the ball		
Pass/Fail Condition: N/A		
Pre-Conditions: CV Subsystem successfully connects to camera, ball is being detected Input: Ball is lifted upwards		
Expected Results: CV tracks the ball in motion	Actual Results: As expected	
Post-Conditions: N/A		

Table 8: Ball Upward Detection Test

Test ID: CV3	**Ball Downward Detection Test**	Status: PASS
Description: Tests if CV Subsystem can detect downward motion of the ball		
Pass/Fail Condition: N/A		
Pre-Conditions: CV Subsystem successfully connects to camera, ball is being detected Input: Ball is moved downward		
Expected Results: CV tracks the ball in motion	Actual Results: As expected	
Post-Conditions: N/A		

Table 9: Ball Downward Detection Test

Test ID: CV4 **Ball Rightward Detection Test**	Status: PASS
Description: Tests if CV Subsystem can detect rightward motion of the ball	
Pass/Fail Condition: N/A	
Pre-Conditions: CV Subsystem successfully connects to camera, ball is being detected Input: Ball is moved rightwards	
Expected Results: CV tracks the ball in motion	Actual Results: As expected
Post-Conditions: N/A	

Table 10: ball upward detection Test

Test ID: CV5 **Ball Leftward Detection Test**	Status: PASS
Description: Tests if CV Subsystem can detect leftward motion of the ball	
Pass/Fail Condition: N/A	
Pre-Conditions: CV Subsystem successfully connects to camera, ball is being detected Input: Ball is moved leftward	
Expected Results: CV tracks the ball in motion	Actual Results: As expected
Post-Conditions: N/A	

Table 11: ball upward detection Test

Test ID: CV6	**CV Timeout Test**	Status: PASS
Description: Tests if CV subsystem times out		
Pass/Fail Condition: Times out within 1 second of initiation		
Pre-Conditions: CV is in state 1 Input: N/A		
Expected Results: Times out in 8 seconds	Actual Results: As expected	
Post-Conditions: cvmodule is closed		

Table 12: cvmodule timeout test

Test ID: CV7	**CV Transition: State 1 to 2**	Status: PASS
Description: Tests that the CV state transitions from state 1 to 2 when ball is moving away from player		
Pass/Fail Condition: State changes within 0.5 seconds of real-time		
Pre-Conditions: CV is in state 1 Input: Ball is moving towards system-side		
Expected Results: CV moves to state 2	Actual Results: As expected	
Post-Conditions: CV is in state 2		

Table 13: CV Transition: State 1 to 2

Test ID: CV8	**CV Transition: State 2 to 3**	Status: PASS
Description: Tests that the CV state transitions from state 2 to 3 when ball is descending		
Pass/Fail Condition: State changes within 0.5 seconds of real-time		
Pre-Conditions: CV is in state 2 Input: Ball is moved downward in frame		
Expected Results: CV moves to state 3	Actual Results: As expected	
Post-Conditions: CV is in state 3		

Table 14: CV Transition: State 2 to 3

Test ID: CV9	**CV Transition: State 3 to 0**	Status: PASS
Description: Tests that the CV state transitions from state 3 to 0 when ball is ascending		
Pass/Fail Condition: State changes within 0.5 seconds of real-time		
Pre-Conditions: CV is in state 3 Input: Ball is moved upward in frame		
Expected Results: CV moves to state 0	Actual Results: As expected	
Post-Conditions: CV is in state 0		

Table 15: CV Transition: State 3 to 0

Test ID: CV10 **Sends Hit Signal to SmartServe** Status: PASS	
Description: Tests that the CV Subsystem sends a "GOOD" signal to SmartServe	
Pass/Fail Condition: N/A	
Pre-Conditions: CV is in state 3 Input: Ball is moved upward in frame	
Expected Results: "GOOD" signal sent to SmartServe	Actual Results: As expected
Post-Conditions: CV is in state 0	

Table 16: good signal test

4.2.2 ShotRecommender

Test ID: SR1 **ShotRecommender Listen Test** Status: PASS	
Description: The ShotRecommender service responds to HTTP calls on port 8080.	
Pass/Fail Condition: The system waits until a request.	
Pre-Conditions: N/A Input: None	
Expected Results: N/A	Actual Results: As expected
Post-Conditions: N/A	

Table 17: ShotRecommender Listen Test

Test ID: SR2 **ShotRecommender Query Test** Status: FAIL	
Description: The ShotRecommender calls the “query” method for user data.	
Pass/Fail Condition: The call returns a table of user performance data.	
Pre-Conditions: The SQL database is running on port 3306. Input: a valid user id for the “performance” procedure	
Expected Results: table of data	Actual Results: Procedure not found
Post-Conditions: N/A	

Table 18: ShotRecommender Query Test

Test ID: SR3 **ShotRecommender Random Shot Test** Status: PASS	
Description: The ShotRecommender receives a request for a shot.	
Pass/Fail Condition: The service generates a random shot.	
Pre-Conditions: The service is running on port 8080. Input: an HTTP request with “Random” as the mode parameter	
Expected Results: a random shot which adheres to requirements	Actual Results: As expected
Post-Conditions: N/A	

Table 19: ShotRecommender Random Shot Test

Test ID: SR4	**ShotRecommender Training Shot Test**	Status: FAIL
Description: The ShotRecommender receives a request for a shot.		
Pass/Fail Condition: The service generates a shot.		
Pre-Conditions: The service is running on port 8080. Input: an HTTP request with “Train” as the mode parameter		
Expected Results: a random shot which adheres to requirements	Actual Results: Model not found	
Post-Conditions: N/A		

Table 20: ShotRecommender Training Shot Test

Test ID: SR5	**ShotRecommender UpdateModel Test**	Status: FAIL
Description: The ShotRecommender receives a status update for a shot.		
Pass/Fail Condition: The service changes the model in response.		
Pre-Conditions: The service is running on port 8080. Input: an HTTP request with the shot id and returned boolean as parameters.		
Expected Results: the model is updated	Actual Results: Model not found	
Post-Conditions: N/A		

Table 21: ShotRecommender UpdateModel Test

4.2.3 Shooting Model

Test ID: SM1 **ShootingModel calculateYawAngle Test** Status: PASS	
Description: The calculateYawAngle method returns an accurate yaw angle in degrees.	
Pass/Fail Condition: The method returns an angle in degrees accurate to a whole number.	
Pre-Conditions: N/A Input: xDist, yDist; distance to desired shot's x-coordinate and y-coordinate.	
Expected Results: A yaw angle in degrees, accurate to a whole number.	Actual Results: As expected
Post-Conditions: N/A	

Table 22: ShootingModel calculateYawAngle Test

Test ID: SM2 **ShootingModel calculateVelocity Test** Status: PASS	
Description: The calculateVelocity method returns an accurate velocity in meters/second.	
Pass/Fail Condition: The method returns the velocity accurate to a whole number.	
Pre-Conditions: N/A Input: N/A	
Expected Results: Velocity in m/s, accurate to a whole number.	Actual Results: As expected
Post-Conditions: N/A	

Table 23: ShootingModel calculateVeolocty Test

Test ID: SM3 **ShootingModel netHeightChecker Test** Status: PASS	
Description: ThenetHeightChecker method checks whether the desired shot will pass over the net.	
Pass/Fail Condition: The method returns the correct boolean indicating if the shot will pass over the net.	
Pre-Conditions: N/A Input: N/A.	
Expected Results: A boolean; True is shot will pass over the net, False otherwise.	Actual Results: As expected
Post-Conditions: N/A	

Table 24: ShootingModel netHeightChecker Test

4.2.4 Data Storage

Test ID: DS1 **Data Storage Sign Up Test** Status: PASS	
Description: Data Storage receives user name and password to sign up	
Pass/Fail Condition: User table updates with correct parameters	
Pre-Conditions: The SQL database is running on port 3306. Input: User name and user password	
Expected Results: User table updated	Actual Results: As expected
Post-Conditions: N/A	

Table 25: Data Storage Sign Up Test

Test ID: DS2 **Data Storage Next Shot Test** Status: PASS	
Description: Data Storage returns a shot type for the system to execute	
Pass/Fail Condition: a specified desired zone the system must aim for is returned	
Pre-Conditions: The SQL database is running on port 3306. Input: Desired zone id as an integer	
Expected Results: Return valid shot parameters, speed and angular velocity	Actual Results: As expected
Post-Conditions: N/A	

Table 26: Data Storage Next Shot Test

Test ID: DS3 **Data Storage Returned Test** Status: PASS	
Description: Data Storage received parameters for a successful or missed shot	
Pass/Fail Condition: Returnrate table updates with correct parameters	
Pre-Conditions: The SQL database is running on port 3306. Input: Timestamp, user and shot ids	
Expected Results: Returnrate table updated	Actual Results: As expected
Post-Conditions: N/A	

Table 27: Data Storage Returned Test

Test ID: DS4 **Data Storage Sign In Test** Status: PASS	
Description: Data Storage receives user name and password and authenticates it	
Pass/Fail Condition: Returns accurate boolean value according to matching of user name and password	
Pre-Conditions: The SQL database is running on port 3306. Input: User name and user password	
Expected Results: True is returned if the password matches the user name, false if they do not match	Actual Results: As expected
Post-Conditions: N/A	

Table 28: Data Storage Sign In Test

4.2.5 User Interface

Test ID: UI1 **User Interface Display Test** Status: PASS	
Description: All elements of UI are displayed in a window	
Pass/Fail Condition: UI displays when program is run	
Pre-Conditions:N/A Input: N/A	
Expected Results: Window opens with Welcome Screen	Actual Results: As expected
Post-Conditions: Application running	

Table 29: User Interface Display Test

Test ID: UI2 **User Interface Button Test** Status: PASS	
Description: All buttons should do some action when pressed	
Pass/Fail Condition: When pressed, buttons change the state of the application and return	
Pre-Conditions: Application UI is running Input: N/A	
Expected Results: Return true when button is pressed	Actual Results: As expected
Post-Conditions: ??	

Table 30: User Interface Button Test

Test ID: UI3 **User Interface Mode Test** Status: PASS	
Description: Mode should be assigned when it is picked in a dropdown	
Pass/Fail Condition: Mode variable assigned selected value	
Pre-Conditions: Application UI is running Input: N/A	
Expected Results: Return value selected	Actual Results: As expected
Post-Conditions: N/A	

Table 31: User Interface Mode Test

Test ID: UI4 **User Interface Sign up Test** Status: FAIL	
Description: When signup button is pressed, user is added with given parameters	
Pass/Fail Condition: User inputs are sent to Data Storage	
Pre-Conditions: Application UI is running Input: User name and user password	
Expected Results: Return user parameters	Actual Results: Page not implemented
Post-Conditions: N/A	

Table 32: User Interface Sign up Test

4.2.6 SmartServe

Test ID: SS1 **ShotRecommendation Connection Test - Pass** Status: PASS	
Description: The ShotRecommendation class will call the *connect* method with port 8080 as a parameter.	
Pass/Fail Condition: The method should return true.	
Pre-Conditions: The ShotRecommendation server is running on port 8080. Input: 8080	
Expected Results: true	Actual Results: As expected
Post-Conditions: N/A	

Table 33: ShotRecommendation Connection Test - Pass

Test ID: SS2 **ShotRecommendation Connection Test - Fail** Status: PASS	
Description: The ShotRecommendation class will call the *connect* method with port 8090 as a parameter.	
Pass/Fail Condition: The method should return false.	
Pre-Conditions: The ShotRecommendation server is running on port 8080. Input: 8090	
Expected Results: false	Actual Results: As expected
Post-Conditions: N/A	

Table 34: ShotRecommendation Connection Test - Fail

Test ID: SS3 **ShotRecommendation Request Shot - Random** Status: PASS	
Description: The ShotRecommendation class will call the *getRecommendation* method with Random mode as a parameter.	
Pass/Fail Condition: The method should a random shot of the form “X=A.BC,Y=A.BC,V=A.BC,W=A.BC” where the values are within the requirements of the system.	
Pre-Conditions: The ShotRecommendation server is running on port 8080. Input: Mode.Random	
Expected Results: A valid shot, in string form.	Actual Results: As expected
Post-Conditions: N/A	

Table 35: ShotRecommendation Request Shot - Random

Test ID: SS4 **ShotRecommendation Request Shot - Train** Status: FAIL	
Description: The ShotRecommendation class will call the *getRecommendation* method with Training mode as a parameter.	
Pass/Fail Condition: The method should a shot of the form "X=A.BC,Y=A.BC,V=A.BC,W=A.BC" where the values are within the requirements of the system.	
Pre-Conditions: The ShotRecommendation server is running on port 8080. Input: Mode.Train	
Expected Results: A valid shot, in string form.	Actual Results: Model not found
Post-Conditions: N/A	

Table 36: ShotRecommendation Request Shot - Train

Test ID: SS5 **ShotRecommendation Request Shot - One-shot** Status: PASS	
Description: The ShotRecommendation class will call the *getRecommendation* method with One-shot mode as a parameter.	
Pass/Fail Condition: The method should a shot of the form "X=A.BC,Y=A.BC,V=A.BC,W=A.BC" where the values are within the requirements of the system. Repeated requests should return the same shot.	
Pre-Conditions: The ShotRecommendation server is running on port 8080. Input: Mode.OneShot	
Expected Results: A valid shot, in string form.	Actual Results: As expected
Post-Conditions: N/A	

Table 37: ShotRecommendation Request Shot - One-shot

Test ID: SS6 **ShotRecommendation Model Update** Status: FAIL	
Description: The ShotRecommendation class will call the *updateModel* method.	
Pass/Fail Condition: The ShotRecommender does not throw an error and the model is updated.	
Pre-Conditions: The ShotRecommendation server is running on port 8080. Input: a previously request shot and false	
Expected Results: N/A	Actual Results: Model not found
Post-Conditions: N/A	

Table 38: ShotRecommendation Model Update

Test ID: SS7 **ShotRecommendation Model Update** Status: FAIL	
Description: The ShotRecommendation class will call the *updateModel* method.	
Pass/Fail Condition: The ShotRecommender does not throw an error and the model is updated.	
Pre-Conditions: The ShotRecommendation server is running on port 8080. Input: a previously request shot and true	
Expected Results: N/A	Actual Results: Model not found
Post-Conditions: N/A	

Table 39: ShotRecommendation Request Shot - One-shot

Test ID: SS8 **ComputerVisionController Connection Test - Pass** Status: PASS	
Description: The CV class will call the *connection* method.	
Pass/Fail Condition: The method returns true.	
Pre-Conditions: The CV server is running on port 8000. Input: 8000	
Expected Results: true	Actual Results: As expected
Post-Conditions: N/A	

Table 40: ComputerVisionController Connection Test - Pass

Test ID: SS9 **ComputerVisionController Connection Test - Fail** Status: PASS	
Description: The CV class will call the *connection* method.	
Pass/Fail Condition: The method returns false.	
Pre-Conditions: The CV server is running on port 8001. Input: 8000	
Expected Results: false	Actual Results: As expected
Post-Conditions: N/A	

Table 41: ComputerVisionController Connection Test - Fail

Test ID: SS10 **ComputerVisionController Detect Test - Timeout** Status: PASS	
Description: The CV class will call the *start* method and no ball is introduced into the frame.	
Pass/Fail Condition: The method returns false.	
Pre-Conditions: The CV server is running on port 8000. Input: N/A	
Expected Results: false	Actual Results: As expected
Post-Conditions: N/A	

Table 42: ComputerVisionController Detect Test - Timeout

Test ID: SS11 **ComputerVisionController Detect Test - Detect** Status: PASS	
Description: The CV class will call the *start* method and a ball is introduced into the frame, emulating a successful shot.	
Pass/Fail Condition: The method returns true.	
Pre-Conditions: The CV server is running on port 8000. Input: N/A	
Expected Results: true	Actual Results: As expected
Post-Conditions: N/A	

Table 43: ComputerVisionController Detect Test - Detect

Test ID: SS12 **SQLConnector Connection Test - Pass** Status: PASS	
Description: The SQLConnector class will call the *connect* method.	
Pass/Fail Condition: The method returns true.	
Pre-Conditions: The SQL server is running on port 3306. Input: 3306	
Expected Results: true	Actual Results: As expected
Post-Conditions: N/A	

Table 44: SQLConnector Connection Test - Pass

Test ID: SS13 **SQLConnector Connection Test - Fail** Status: PASS	
Description: The SQLConnector class will call the *connect* method.	
Pass/Fail Condition: The method returns false.	
Pre-Conditions: The SQL server is not running. Input: 3306	
Expected Results: false	Actual Results: As expected
Post-Conditions: N/A	

Table 45: SQLConnector Connection Test - Pass

Test ID: SS14 **SQLConnector Query Test - Signup** Status: PASS	
Description: The SQLConnector class will call the *save* method.	
Pass/Fail Condition: The user is saved in the database.	
Pre-Conditions: The SQL server is running on port 3306. Input: Object[] with “Chris” and “password123” for the “sign_up” procedure	
Expected Results: true	Actual Results: As expected
Post-Conditions: N/A	

Table 46: SQLConnector Query Test - Signup

Test ID: SS15 **SQLConnector Query Test - Returned** Status: PASS	
Description: The SQLConnector class will call the *save* method.	
Pass/Fail Condition: The shot is saved in the database correctly.	
Pre-Conditions: The SQL server is running on port 3306. Input: Object[] with 25, 1, 1 and the current time for the “returned” procedure	
Expected Results: true	Actual Results: As expected
Post-Conditions: N/A	

Table 47: SQLConnector Query Test - Returned

Test ID: SS16 **SQLConnector Query Test - Login** Status: PASS	
Description: The SQLConnector class will call the *save* method.	
Pass/Fail Condition: The shot is saved in the database correctly.	
Pre-Conditions: The SQL server is running on port 3306. Input: Object[] with “Chris” and “password123” for the “login” procedure	
Expected Results: true	Actual Results: As expected
Post-Conditions: N/A	

Table 48: SQLConnector Query Test - Returned

Test ID: SS17 **ArduinoController Connection Test - Pass** Status: PASS	
Description: The ArduinoController class will call the *test* method, repeat for all arduinos.	
Pass/Fail Condition: The method returns true.	
Pre-Conditions: The Arduinos are plugged in via USB and loaded with the correct code. Input: the port for the Arduino(s)	
Expected Results: true	Actual Results: As expected
Post-Conditions: N/A	

Table 49: ArduinoController Connection Test - Pass

Test ID: SS18 **ArduinoController Connection Test - Fail** Status: PASS	
Description: The ArduinoController class will call the *test* method, repeat for all arduinos.	
Pass/Fail Condition: The method returns false.	
Pre-Conditions: The Arduinos are plugged in via USB and loaded with the correct code. Input: “some-test-string”	
Expected Results: false	Actual Results: As expected
Post-Conditions: N/A	

Table 50: ArduinoController Connection Test - Fail

Test ID: SS19 **ArduinoController Test - Pan** Status: PASS	
Description: The ArduinoController class will call the *shoot* method.	
Pass/Fail Condition: The system pans to 75 degrees.	
Pre-Conditions: The Arduinos are plugged in via USB and loaded with the correct code. Input: a ShotDetail object with pan of 75 degrees	
Expected Results: true	Actual Results: As expected
Post-Conditions: N/A	

Table 51: ArduinoController Test - Pan

Test ID: SS20 **ArduinoController Test - Pan** Status: PASS	
Description: The ArduinoController class will call the *shoot* method.	
Pass/Fail Condition: The system pans to 120 degrees.	
Pre-Conditions: The Arduinos are plugged in via USB and loaded with the correct code. Input: a ShotDetail object with pan of 120 degrees	
Expected Results: true	Actual Results: As expected
Post-Conditions: N/A	

Table 52: ArduinoController Test - Pan

Test ID: SS21 **ArduinoController Test - Shoot** Status: PASS	
Description: The ArduinoController class will call the *shoot* method.	
Pass/Fail Condition: The system shoots at 75% power.	
Pre-Conditions: The Arduinos are plugged in via USB and loaded with the correct code. Input: 75	
Expected Results: true	Actual Results: As expected
Post-Conditions: N/A	

Table 53: ArduinoController Test - Shoot

Test ID: SS22	**ArduinoController Test - Shoot**	Status: PASS
Description: The ArduinoController class will call the *shoot* method.		
Pass/Fail Condition: The system shoots at 100% power.		
Pre-Conditions: The Arduinos are plugged in via USB and loaded with the correct code. Input: 100		
Expected Results: true	Actual Results: As expected	
Post-Conditions: N/A		

Table 54: ArduinoController Test - Shoot

5 Test Case-Requirement Traceability Matrix

Table 55: Matrix to Match Tests to Functional Requirements [1]

Functional Requirement-Test Matrix																		
	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16	F17	F18
SMC1	X	X	X	X	X													X
SMC2	X	X			X													X
SMC3	X			X														
SMC4	X																	
SMC5	X		X															
CV1					X													X
CV2					X													
CV3					X													
CV4					X													X
CV5					X													X
CV6						X												X
CV7					X													
CV8					X													
CV9					X													
CV10						X												
SR1														X	X			
SR2							X							X	X			
SR3																		
SR4														X				
SR5														X				
SM1	X		X		X													
SM2	X	X			X													X
SM3																		
DS1								X										
DS2	X	X	X	X	X	X												X
DS3						X												
DS4									X									
UI1								X	X	X	X	X	X			X	X	
UI2								X	X	X	X	X	X			X	X	
UI3																X		
UI4								X										

Table 56: Matrix to Match Tests to Functional Requirements [2]

Functional Requirement-Test Matrix																		
	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16	F17	F18
SS1							X							X	X			
SS2							X							X	X			
SS3																		
SS4														X				
SS5															X			
SS6														X				
SS7														X				
SS8					X	X												X
SS9					X	X												X
SS10					X	X												X
SS11					X	X												X
SS12						X		X	X				X					X
SS13						X		X	X				X					X
SS14								X										
SS15						X												
SS16									X									
SS17	X	X	X	X	X													X
SS18	X	X	X	X	X													X
SS19	X		X															
SS20	X		X															
SS21	X	X			X													X
SS22	X	X			X													X

Table 57: Matrix to Match Tests to Non-Functional Requirements [1]

Non-Functional Requirement-Test Matrix																		
	LF1	UH1	UH2	P1	P2	P4	P5	OE2	MS2	S1	S2	P1	LC1	HS1	HS2	HS3	HS4	HS5
SMC1														X	X			
SMC2														X	X			
SMC3														X				
SMC4														X				
SMC5														X				
CV1								X										
CV2																		
CV3																		
CV4																		
CV5																		
CV6																		
CV7																		
CV8																		
CV9																		
CV10																		
SR1														X	X			
SR2														X				
SR3														X				
SR4																		
SR5																		
SM1														X				
SM2														X	X			
SM3														X				
DS1						X				X								
DS2														X	X			
DS3					X													
DS4																		
UI1	X	X	X		X	X	X		X	X		X						
UI2	X	X	X	X														
UI3			X	X														
UI4			X			X				X								

Table 58: Matrix to Match Tests to Non-Functional Requirements [2]

Non-Functional Requirement-Test Matrix																		
	LF1	UH1	UH2	P1	P2	P4	P5	OE2	MS2	S1	S2	P1	LC1	HS1	HS2	HS3	HS4	HS5
SS1															X			
SS2															X			
SS3															X			
SS4															X			
SS5															X			
SS6																		
SS7																		
SS8																		
SS9																		
SS10																		
SS11																		
SS12					X	X				X								
SS13					X	X				X				X	X			
SS14						X				X				X	X			
SS15					X													
SS16																		
SS17														X	X			
SS18														X	X			
SS19														X				
SS20														X				
SS21														X	X			
SS22														X	X			