

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	31

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

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: ????
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/powerd on.		
Input: Integer indicating the amount of degrees to be rotated.		
Expected Results: Feeding mechanism rotates to the required position.		Actual Results: ????
Post-Conditions: N/A		

Table 2: Feeding Mechanism Rotation Test

Test ID: SMC2	Shooting Control Test	Status: ????
Description: The shooting control shoots the ball via a rotating wheel at the requested power level to achieve the desired speed.		
Pass/Fail Condition: ???		
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: ????		Actual Results: ????
Post-Conditions: N/A		

Table 3: Shooting Control Test

Test ID: SMC3	Four Position Roll Test	Status: ????
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: ????
Post-Conditions: N/A		

Table 4: Four Position Roll Test

Test ID: SMC4	Adjustable Pitch Control Test	Status: ????
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: ????
Post-Conditions: N/A		

Table 5: Adjustable Pitch Control Test

Test ID: SMC5 Panning Shooting Mechanism Across the Table Status: ????	
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: ????
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: ????	
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: ????
Post-Conditions: N/A	

Table 7: Ball Detection Test

Test ID: CV2	Ball Upward Detection Test	Status: ????
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: ????	
Post-Conditions: N/A		

Table 8: ball upward detection Test

Test ID: CV3	Ball Downward Detection Test	Status: ????
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: ????	
Post-Conditions: N/A		

Table 9: ball upward detection Test

Test ID: CV4	Ball Rightward Detection Test	Status: ????
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: ????	
Post-Conditions: N/A		

Table 10: ball upward detection Test

Test ID: CV5	Ball Leftward Detection Test	Status: ????
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: ????	
Post-Conditions: N/A		

Table 11: ball upward detection Test

Test ID: CV6	CV Timeout Test	Status: ????
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: ????
Post-Conditions: cvmodule is closed		

Table 12: cvmodule timeout test

Test ID: CV7	CV Transition: State 1 to 2	Status: ????
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: ????
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: ????
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: ????	
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: ????
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: ????	
Post-Conditions: CV is in state 0		

Table 15: CV Transition: State 3 to 0

Test ID: CV10	Sends Hit Signal to SmartServe	Status: ????
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: ????
Post-Conditions: CV is in state 0		

Table 16: good signal test

4.2.2 ShotRecommender

Test ID: SR1	ShotRecommender Listen Test	Status: ????
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: ????
Post-Conditions: N/A		

Table 17: ShotRecommender Listen Test

Test ID: SR2	ShotRecommender Query Test	Status: ????
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: ????	
Post-Conditions: N/A		

Table 18: ShotRecommender Query Test

Test ID: SR3	ShotRecommender Random Shot Test	Status: ????
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: ????	
Post-Conditions: N/A		

Table 19: ShotRecommender Random Shot Test

Test ID: SR4	ShotRecommender Training Shot Test	Status: ????
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: ????
Post-Conditions: N/A		

Table 20: ShotRecommender Training Shot Test

Test ID: SR5	ShotRecommender UpdateModel Test	Status: ????
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: ????	
Post-Conditions: N/A		

Table 21: ShotRecommender UpdateModel Test

4.2.3 Shooting Model

Test ID: SM1	ShootingModel calculateYawAngle Test	Status: ????
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: ????
Post-Conditions: N/A		

Table 22: ShootingModel calculateYawAngle Test

Test ID: SM2	ShootingModel calculateVelocity Test	Status: ????
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: ????
Post-Conditions: N/A		

Table 23: ShootingModel calculateVelocity Test

Test ID: SM3	ShootingModel netHeightChecker Test	Status: ????
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: ????
Post-Conditions: N/A		

Table 24: ShootingModel netHeightChecker Test

4.2.4 Data Storage

Test ID: DS1	Data Storage Sign Up Test	Status: ????
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: ????
Post-Conditions: ????		

Table 25: Data Storage Sign Up Test

Test ID: DS2	Data Storage Next Shot Test	Status: ????
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: ????
Post-Conditions: ????		

Table 26: Data Storage Next Shot Test

Test ID: DS3	Data Storage Returned Test	Status: ????
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: ????
Post-Conditions: ????		

Table 27: Data Storage Returned Test

Test ID: DS4	Data Storage Sign In Test	Status: ????
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: ????	
Post-Conditions: ????		

Table 28: Data Storage Sign In Test

4.2.5 User Interface

Test ID: UI1	User Interface Display Test	Status: Complete
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: Window opens with Welcome Screen	
Post-Conditions: Application running		

Table 29: User Interface Display Test

Test ID: UI2	User Interface Button Test	Status: ????
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: ??	
Post-Conditions: ??		

Table 30: User Interface Button Test

Test ID: UI3	User Interface Mode Test	Status: ????
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: ??	
Post-Conditions: ??		

Table 31: User Interface Mode Test

4.2.6 SmartServe

Test ID: SR1	ShotRecommendation Connection Test - Pass	Status: ????
Description: The ShotRecommendation class will call the <i>connect</i> 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: ????	
Post-Conditions: N/A		

Table 32: ShotRecommendation Connection Test - Pass

Test ID: SR2	ShotRecommendation Connection Test - Fail	Status: ????
Description: The ShotRecommendation class will call the <i>connect</i> 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: ????	
Post-Conditions: N/A		

Table 33: ShotRecommendation Connection Test - Fail

Test ID: SR3 ShotRecommendation Request Shot - Random Status: ????	
Description: The ShotRecommendation class will call the <i>getRecommendation</i> 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: ????
Post-Conditions: N/A	

Table 34: ShotRecommendation Request Shot - Random

Test ID: SR4 ShotRecommendation Request Shot - Train Status: ????	
Description: The ShotRecommendation class will call the <i>getRecommendation</i> 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: ????
Post-Conditions: N/A	

Table 35: ShotRecommendation Request Shot - Train

Test ID: SS1	ShotRecommendation Request Shot - One-shot	Status: ????
Description: The ShotRecommendation class will call the <i>getRecommendation</i> 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: ????	
Post-Conditions: N/A		

Table 36: ShotRecommendation Request Shot - One-shot

Test ID: SS2	ShotRecommendation Model Update	Status: ????
Description: The ShotRecommendation class will call the <i>updateModel</i> 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: ????	
Post-Conditions: N/A		

Table 37: ShotRecommendation Model Update

Test ID: SS3	ShotRecommendation Model Update	Status: ????
Description: The ShotRecommendation class will call the <i>updateModel</i> 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: ????	
Post-Conditions: N/A		

Table 38: ShotRecommendation Request Shot - One-shot

Test ID: SS4	ComputerVisionController Connection Test - Pass	Status: ????
Description: The CV class will call the <i>connection</i> 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: ????	
Post-Conditions: N/A		

Table 39: ComputerVisionController Connection Test - Pass

Test ID: SS5 ComputerVisionController Connection Test - Fail Status: ????	
Description: The CV class will call the <i>connection</i> 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: ????
Post-Conditions: N/A	

Table 40: ComputerVisionController Connection Test - Fail

Test ID: SS6 ComputerVisionController Detect Test - Timeout Status: ????	
Description: The CV class will call the <i>start</i> 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: ????
Post-Conditions: N/A	

Table 41: ComputerVisionController Detect Test - Timeout

Test ID: SS7 ComputerVisionController Detect Test - Detect Status: ????	
Description: The CV class will call the <i>start</i> 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: ????
Post-Conditions: N/A	

Table 42: ComputerVisionController Detect Test - Detect

Test ID: SS8 SQLConnector Connection Test - Pass Status: ????	
Description: The SQLConnector class will call the <i>connect</i> 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: ????
Post-Conditions: N/A	

Table 43: SQLConnector Connection Test - Pass

Test ID: SS9	SQLConnector Connection Test - Fail		Status: ????
Description: The SQLConnector class will call the <i>connect</i> method.			
Pass/Fail Condition: The method returns false.			
Pre-Conditions: The SQL server is not running.			
Input: 3306			
Expected Results: false		Actual Results: ????	
Post-Conditions: N/A			

Table 44: SQLConnector Connection Test - Pass

Test ID: SS10	SQLConnector Query Test - Signup	Status: ????
Description: The SQLConnector class will call the <i>save</i> 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: ????
Post-Conditions: N/A		

Table 45: SQLConnector Query Test - Signup

Test ID: SS11	SQLConnector Query Test - Returned	Status: ????
Description: The SQLConnector class will call the <i>save</i> 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: ????
Post-Conditions: N/A		

Table 46: SQLConnector Query Test - Returned

Test ID: SS12	SQLConnector Query Test - Login	Status: ????
Description: The SQLConnector class will call the <i>save</i> 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: ????
Post-Conditions: N/A		

Table 47: SQLConnector Query Test - Returned

Test ID: SS13	ArduinoController Connection Test - Pass	Status: ????
Description: The ArduinoController class will call the <i>test</i> 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: ????	
Post-Conditions: N/A		

Table 48: ArduinoController Connection Test - Pass

Test ID: SS14	ArduinoController Connection Test - Fail	Status: ????
Description: The ArduinoController class will call the <i>test</i> 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: ????	
Post-Conditions: N/A		

Table 49: ArduinoController Connection Test - Fail

Test ID: SS15	ArduinoController Test - Pan	Status: ????
Description: The ArduinoController class will call the <i>shoot</i> 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: ????	
Post-Conditions: N/A		

Table 50: ArduinoController Test - Pan

Test ID: SS16	ArduinoController Test - Pan	Status: ????
Description: The ArduinoController class will call the <i>shoot</i> 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: ????	
Post-Conditions: N/A		

Table 51: ArduinoController Test - Pan

Test ID: SS17	ArduinoController Test - Shoot	Status: ????
Description: The ArduinoController class will call the <i>shoot</i> 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: ????	
Post-Conditions: N/A		

Table 52: ArduinoController Test - Shoot

Test ID: SS18	ArduinoController Test - Shoot	Status: ????
Description: The ArduinoController class will call the <i>shoot</i> 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: ????	
Post-Conditions: N/A		

Table 53: ArduinoController Test - Shoot

5 Test Case-Requirement Traceability Matrix

Table 54: Matrix to Match Tests to Functional Requirements

[illegible]

Table 55: Matrix to Match Tests to Non-Functional Requirements

[illegible]