**ONEWEEK** Scenario Spec

Spot Me

Craig Loewen, Akanksha Bansal, Jhony Guan

**Spec Status:** Draft

# Scenario Description

## Elevator Pitch / Narrative

Amy wants to start working out, but has never done it before and has no clue on where to start, or how to do any exercises. She googles a workout plan with some basic weight lifting exercises, but wants to do them correctly, and ensure that she doesn’t hurt herself.

Amy downloads Spot Me and opens the exercises that she wants to learn and practice on. The app tells her specifics on how to improve her form, and Amy finds it helps her learn how to work out confidently. Even after she has been working out for two years she still uses the app to help ensure that her form is good on different exercises.

## Target User or System

Beginner and Experienced Weightlifters

Possible use but not target: Physiotherapist offices

## Problem Statement and Supporting Customer Insights

People have a hard time getting motivated to go to the gym and work out. A big reason is: ‘I’ve never done it before and I don’t know how to do anything.’ A personal trainer can be expensive, and therefore isn’t a possibility for all users. This application will teach people how to do the basics properly, and give them confidence to start their fitness journey. As well, experienced users can use the app to check their form, such as when they are doing a new personal record for their squat and want to make sure they went down to 90 degrees.

## Existing Solutions or Expectations

To learn a new exercise a user can watch a video online and then practice in front of the mirror. Or, they could have an experienced person watch them do the exercise and give advice. This application will seek to mirror the experience seen when an experienced person is giving advice on how to do an exercise.

# Requirements

## Functional Requirements

|  |  |  |
| --- | --- | --- |
| No. | Requirement | Priority |
| 1 | Create a system to add/remove/view exercises | 1 |
| 2 | Allow users to see their last set and take advice from it | 1 |
| 3 | Identify correct or incorrect form on a given skeleton in an exercise | 1 |
| 4 | Give generalized advice on how to fix incorrect form | 1 |
| 5 | Give customized advice on how to fix incorrect form | 1 |
| 6 | An engaging first-time User Experience for the app | 2 |
| 7 | Create a website to learn more about the app and download it | 2 |
| 8 | An engaging first-time User Experience when opening an exercise | 3 |
| 9 | Have an option to start a workout routine | 3 |

## Other Deliverables

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Deliverables | Priority | Due Date |
| 1 | Working product and demo due for presentation at the Science Fair | 1 | July 27 1PM |
| 2 | 1 to 3 minute video that describes/demonstrates the project | 1 | July 26 11:59PM |
| 3 | Web Page for info about Spot Me | 2 | July 24 |

# Front End Design

The front end design for this project is left totally open. Some already exists on [www.craigloewen.com/spot-me](http://www.craigloewen.com/spot-me) but this is old, and was a prototype. That design is open source so feel free to sample and use it, but do not feel tied to it in anyway.

Figure 1 below shows a rough idea of the user flow for the app. The black solid arrows are the Minimal Viable Product (MVP), and the dotted lines are features that would be nice to add, while the different colors symbolize different flows of using the app.

Ink Drawings
Ink Drawings
Ink Drawings
Ink Drawings
Ink Drawings
Ink Drawings
Ink Drawings
Ink Drawings
Ink Drawings
Ink Drawings
Ink Drawings
Ink Drawings
Ink Drawings
￼￼￼
￼￼￼
￼￼￼￼
￼￼￼
￼
￼
￼
￼￼￼￼￼￼￼￼￼￼
￼￼￼￼￼￼￼￼￼
￼￼￼￼
￼￼￼￼￼￼
￼￼￼￼
￼￼￼￼￼￼￼￼￼
￼
￼￼￼
￼
￼￼￼￼￼￼￼￼￼￼￼￼
￼￼￼￼
￼￼￼￼
￼￼
￼
￼￼￼￼￼￼￼￼￼￼￼￼￼￼￼￼￼￼￼￼￼￼￼￼￼￼
￼￼￼￼￼
￼
￼￼￼￼￼
￼￼￼￼￼￼￼￼￼￼￼￼￼￼￼
￼
￼
￼
￼
￼￼￼￼￼￼￼￼￼￼￼￼￼￼￼￼￼￼￼￼￼￼￼￼￼
￼￼￼
￼￼￼
￼
￼
￼￼￼
￼
￼
￼
Ink Drawings
Ink Drawings
Ink Drawings
Ink Drawings
Ink Drawings
Ink Drawings
Ink Drawings
Ink Drawings
Ink Drawings
Ink Drawings
￼
￼
￼￼￼￼￼
￼￼￼￼￼￼
￼￼￼￼￼￼￼￼￼￼￼￼￼￼￼￼￼￼￼￼￼￼￼￼￼￼
￼￼￼
￼


Figure 1: User Flow Diagram

## Required Pages

This leads to the definition of the 4 pages that are required for the minimum viable product:

* List of exercises + exercise management (if this is to be done on the same page)
* In exercise - The screen shown when the person is doing the exercise. Needs minor visual feedback
* Post exercise analysis - Show in some more detail how to improve your form
* Exercise addition - Experience for adding a new exercise

## Creating a new exercise experience

An important aspect of the project is that the instruction for improving your form is based upon data from humans, and is the same instruction that a human would give to you. Below is a rough ordered list of what the experience could be for creating a new exercise definition.

1. 'Show me the perfect contracted form' – Obtain one point of data of ideal contracted form
2. 'Move around within the bounds of what would still constitute good form' – Obtain the ‘cloud’ of data that would still constitute good contracted form
3. 'Show me perfect extended form' – Obtain one point of data of ideal extended form
4. 'Move around within the bounds of what would still constitute good form' – Obtain the ‘cloud’ of data that would still constitute good extended form
5. 'Done the basics! Optionally add some more personalized feedback' – Option to just try it and test it out, or add more personalized feedback if wanted
6. Possible final step: Add some kind of test validation (Do a rep and see if it predicts it correctly)

# Back End Design

This project will be coded on Windows Presentation Foundation (WPF) and in C#. It will use a Support Vector Machine to classify correct or incorrect algorithms.

## Why WPF?

This project was initially conceived as a UWP (Universal Windows Platform) app, so that users could use it both on their PCs and on an Xbox. UWP does not fully support skeleton data on the Kinect currently WPF was chosen for this project since it uses a XAML interface, similar to UWP, and in the future it can be used to easily expand into a UWP app.

## File Management System

Each exercise has a certain amount of data that needs to be stored with it. Below is a list of this data:

* Exercise training data (multiple skeletons with class information to train the AI)
  + 24 doubles corresponding to the vector data for the inputted skeleton
  + The class that this skeleton belongs to (so good form, bad form and which type, etc.) these will be stored as integers from 0 to the number of classes
* Classifications Data
  + The number of classifications that this exercise has
  + The names of each classification: ‘Good form – contracted’, ‘Good form – extended’, ‘Bad form – elbows out’, etc.
  + The message that should be shown on screen for each classification ‘Good form!’ ‘Move your elbows in!’ etc.
  + POSSIBLY: Which joint to highlight (Such as wrists or elbows)
  + POSSIBLY: Direction on where to move to make your form better (such as drawing arrows on your elbows pointing up to make you move your arms up

## Function List

Using Figure 1 as the source material, a list of ordered functions is created below. Questions for each function are left in bold.

* User Sees list of exercises
  + Enumerate list of exercises
    - Get exercise name
    - Get exercise good form skeleton for thumbnail(?)
  + Create/Modify/Delete Exercises routines
    - Create/Modify/Delete classifiers routines for each exercise
* Select an exercise
  + Load specific exercise data (including full training data)
  + Train machine learning algorithm based on data
* Start doing reps
  + Detect when a user has paused (I.e: extended or contracted in their rep)
  + Get the form critique for the current identified position
    - **What is this in the form of?**
    - **What should be displayed?**
    - **Should it be text?**
    - **Should it be an arrow pointing to an accepted position?**
  + Save skeletons of each rep to a list to enumerate at the end of use
* Finish their set
* Sees detailed advice on form improvements
  + Enumerate a list of skeletons of all reps done in that set
    - **How to store this data? Should be as a Body or bodyDouble? (most likely body as memory isn’t too much of a concern)**
  + Based on User Input save a skeleton as training data with classifier
    - Include create/modify/delete classifiers here

## API Map

This list of functions is then used to determine a rough structure for the API of the program.

SpotMeML.cs

* Initialize machine learning based on exercise(Exercise inExercise)  
  returns bool for success
* Detect when a user has paused(Body inBody)  
  returns bool for paused or not
* Get form critique for an inputted skeleton(Body inBody)  
  returns the classifier number

Exercise.cs

* Get/Set exercise name [ using a string]
* Get/Set exercise good form (contracted and extended) [using double[] ]
* Get/Set list of classifiers [ using a List<Classifier> ]

Classifier.cs

* Get/set classifier name [ using a string ]

ExerciseFileIO.cs (Could be static?)

* Save Exercise(string exerciseName, exercise inExercise)  
  returns bool for success
* Load Exercise(string exerciseName, out exercise outExercise)  
  returns bool for success
* Delete Exercise(string exerciseName)  
  returns bool for success
* Modify Exercise [Could be done by just giving access to exercise and then saving it]
* Add classifier (Exercise inExercise, Classifier inClassifier)  
  returns bool for success
* Modify classifier(Exercise inExercise, Classifier inClassifier)
* Delete classifier(Exercise inExercise, Classifier inClassifier)  
  returns bool for success
* Add classifier to current exercise (Classifier inClassifier)  
  returns bool for success
* Modify classifier to current exercise (Classifier inClassifier)
* Delete classifier to current exercise (Classifier inClassifier)  
  returns bool for success
* Enumerate Exercise List(out List<string> exerciseNames, out double[][] skeletonData) [Include names, and good form skeleton for thumbnail]  
  returns bool for success
* Get Current Exercise()   
  returns Exercise

MainWindow.cs (The form for displaying the window)

* Display skeleton in small thumbnail fomats(bodyDouble inBodyDouble)  
  returns void

Feedback.cs

* Save skeleton of that rep(Body inBody)  
  returns bool
* Get/Set list of rep skeletons done in that set [ using a List<Body> ]
* POSSIBLE EXTENSION: Only get list of skeletons done with bad form, etc.