SW Engineering CSC648/848

PosFit

Your Virtual Fitness Coach

Prepared by Team 03

1. **Zack Wang** **(zwang14@mail.sfsu.edu)**
2. **Minseon Park** **(mpark5@mail.sfsu.edu)**
3. **John San Jose** **(jsanjose@mail.sfsu.edu)**
4. **Jason Cordis** **(jcordis@mail.sfsu.edu)**
5. **Brian Frey** **(bfrey@mail.sfsu.edu)**
6. **Vincent Wong** **(vwong5@mail.sfsu.edu)**
7. **Darshil Dhameliya** **(ddhameliya@mail.sfsu.edu)**

Milestone 4

Nov. 28, 2021

Revisions	Revision Date	Summary
V1	10/29	First Draft

1) Product summary

PosFit is an AI based web application that targets home gym exercisers. Due to the time and site limitation, many exercisers start looking for a home gym option. The exercisers tend to watch tutorial videos and imitate fitness activities. There are many existing apps or products that can provide such a service, such as YouTube, Mirror, and Keep, etc. The unique service that PosFit offers is to provide feedback of the exercisers' poses or forms. None of our competitors are offering this service in the market.

Our website is: http://netfitness.s3-website-us-west-1.amazonaws.com/

Once users land on the home page, they can click "Search Video" to find the exercises they want to follow. After clicking a video, the users will be taken to the video playing page, and the camera will be turned on automatically to capture the users' movements. Furthermore, users can comment under the video and add the video into their playlist, but these two functionalities can only be done after their login.

If the users are new to the website or not logged in, the button "Login" appears in the header. Both unregistered and registered users can click that button and follow the instructions to either log in their accounts or register accounts. Once the users are logged in, they can upload videos, find their playlist, and change their profiles in the user portal.

There are four main categories of functionality in our product:

- AI Feedback
- User Operations
  - Login
  - Registration
  - Logout
  - Delete Account
- Profile Operations
  - View Profile
  - Edit
  - ChangePassword
- Video Operations
  - Viewing Video
  - Uploading Video
  - Commenting
  - Searchable

2) QA testing:

Mocha: Testing Framework

Chai: Assertion API

Istanbul: Test Coverage Tool

I. Unit Testing

A. User Registration.

1. insertUser():

a) User entries from the registration form are logged into DynamoDB.

B. User Login.

1. retrieveUser():

a) Given user information from DynamoDB including primary keys to specific object keys used for S3 object extraction are pulled from the database.

C. User uploading videos:

1. appendVideoKey():

a) Given a video access key and an email, the recorded key is stored inside the user's upload list.

b) Given a video access key and an email, the recorded key is stored inside the user's playlist.

2. uploadVideo():

a) Given information, ie) email, username and a video object containing video file and video image of thumbnail, video key is appended to user list of uploads and video thumbnail and video file are uploaded onto S3 and metadata stored in DynamoDB.

D. User Account Deletion:

1. deleteUserEntries:

a) Given an existing user email, user's entries in DynamoDB; email/username/bio… etc. and is removed from the list of users in Cognito.

(Parameter Data, Functions and Unit Test Results)

Parameter Data:	Values	Functions:	Results
Email	MochaTester@gmail.com	insertUser(Email, Username,Account Type, Bio, Profile Pic)	Pass
Username	MochaTester	retrieveUser(Email)	Pass
Account type	trainee	appendVideoKey(Email, Video Key, isPlaylist)	Pass
Bio	I am a mocha tester testing our app.	removeVideoKey(Email, Video Key, isPlaylist)	Pass
Profile Pic	None	deleteUserEntries(Email)	Pass
Vide	video: fs.readFileSync("./src/videos/yoga.mp4"), thumbnail: fs.readFileSync("./src/images/Slider/Yoga.jpg"), description: "Yoga test video", title: "MochaTest_uploadVideo", category: "Misc."	uploadVideo(username, email, video)	Pass
isPlaylist	True / False		
Video Key	mochaTestVideo_upload/playlist		

Current Test Coverage:

File	Stmt %	Branch %	Funcs %	Lines %	Uncovered Lines #’s

All Files	0	0	0	0	

II. Integration Test (Tester: John San Jose)

Test Coverage Analysis:
Pass: Tested and Complete
Operational: Untested but meets minimum working req.
Incomplete: Untested but implementation in progress

A. User Account Creation and Deletion.

Steps:		Data:	
1. Go to /register	4. Delete current profile	Email	YellowFruits@gmail.com
2. Create profile		Password	Crunchy1!
3. Go to /delete_account		Result	Pass

B. Users Log In/Out.

Steps:		Data:	
1. Go to /login		Email	YellowFruits@gmail.com
2. Login		Password	Crunchy1!
3. Press “logout” button in navbar		Result	Pass

C. User Video Management.

Steps:		Data	

1. Go to /upload_video	4. Insert a thumbnail and a video file.	video:./src/videos/yoga.mp4, thumbnail:./src/images/Slider/Yoga.jpg, description:Yoga test video, title: MochaTest_uploadVideo`, category:Misc.	
2. Fill upload form with meta-data	5. Submit form		
3. Categorize video		Result	Pass

D. Video Interaction.

Steps:		Data:	
1. Go to /login and login to account	4. Play the video	Email	YellowFruits@gmail.com
2. Go to /search	5. Make a comment	Password	Crunchy1!
3. Click on “banana video” link		Result	Operational

E. Artificial Intelligence/Machine Learning Video Feedback

Steps:		Data:	
1. N/A		N/A	
		Result	Incomplete

3) Code Review:

This code review of the Posfit source code will be primarily based on the Google Javascript style guide which can be found at https://google.github.io/styleguide/jsguide.html. This code review will look at these general rules for Javascript and highlight areas which deviate from Google's guidelines. The code for review was pulled from git@github.com:CSC-648-SFSU/csc648-fa21-04-Team03.git.

Code Formatting:

Google suggests all file names begin with lowercase letters, but the general convention of this project is to begin with uppercase letters, which is fine. Two file names break the convention. The official guidelines suggest a two space indent. The Posfit source code varies between two-space indents and four-space tabs. Additionally, the guidelines call for an 80 character limit to prevent lines of code from being cut off at the end of the monitor. However, the guidelines also suggest using identifiers with clear descriptions and without deleting letters in a word. The source code has a couple of truncated identifiers.

The guidelines also suggest using single quotes to delimit string literals. Posfit has some "" where it could have had a ''. There are various instances of poor block formatting, for example among import statements and in different functions. Also, the guidelines want every statement terminated by a semi-colon. Relying on automatic insertion is forbidden. Posfit is missing some semi-colons. The guidelines want variables declared with const and let, and not with var. Posfit has at least one file using var.

The guidelines don't want the 'new Array()' variadic operator to be called, since it is error prone, instead preferring a literal (search). Another point of divergence between Posfit and the guidelines is about concatenating long strings. Google suggests using template strings instead.

Functions may be nested in other functions, but they should be assigned to a local const. Some of the functions in Posfit violate this guideline. Also, the fat arrow function is preferred over the function keyword, particularly for nested functions. The general functional structure of Posfit has some inconsistencies. Also, method names should generally be verbs or verb phrases, not nouns.

Comments should be included more liberally in the source code, including top level commented overviews of the file. This will help readability and make the files easier to understand. Functions should also have documented descriptions, including parameters and return types.

The whitespace areas of the code could be more consistent, as there are places where the whitespace is uneven. There are many unused imports in the source code, reflecting a project under construction, but they should be removed if not required. The guidelines calls for named exports instead of default exports, but many of the app's exports are default.

Recommendations for improving code:

AWS identification information should be hidden from public facing source code.
The code generally needs more error catching and exception handling than it currently has.
The compiled code has warnings which should be handled.
Current search function needs improvement.
Code should be properly formatted and commented.

DRY principle is violated in some places and some functions can be more effectively reused.
The source code could be better designed to handle testing and debugging.
Security features are lacking and the database is vulnerable to hacking.
Performance efficiency could likely be improved with better database management and handling.
The usability of the site could be improved with a better UI.
The single responsibility principle could be more strictly observed, as some functions have multiple responsibilities.

4) Self-check: Adherence to original Non-functional specifications

Specification	Done	On Track	Issue
Accurate feedback using the AI model			TODO: Connect AI, Test AI
Ease of use			
Fast loading of web pages			
Categorization of exercises		TODO: reduce categories	
Data privacy		TODO: .env file	
Chrome + Firefox + Edge browser compatibility			
Scalable database			
Meaningful variable names using camel style		TODO: Follow Coding Style	
Unified CSS stylesheets		TODO: Fix minor differences in text/styling	
Continuous Integration + Development		TODO: Commit more often, test before push	

Issues:

- Accurate feedback using the AI model -

Our goal is to provide real-time, accurate feedback to the user such as highlighting body parts that do not match the form correctly. However, latency due to computation results in delayed feedback which is likely to reduce the utility of the feedback. The user will most likely be in a different pose by the time the feedback is returned.