CEN 4010 Spring 2018 Milestone 4 Beta Launch

**Team 11’s “HealthFuel”**

**Product Owner**

Justin Rivera

([JustinRivera2021@fau.edu](mailto:JustinRivera2021@fau.edu))

**Scrum Master**

Mohannad Darwish

([Mdarwish2020@fau.edu](mailto:Mdarwish2020@fau.edu))

**Dev Team**

Jean-Patrick Gabon

([jgabon2019@fau.edu](mailto:jgabon2019@fau.edu))

Eric Compbell

([Ecampbell2020@fau.edu](mailto:Ecampbell2020@fau.edu))

Jadyn Frank

([Jadynfrank2020@fau.edu](mailto:Jadynfrank2020@fau.edu))

4/13/23

**Product Summary**

Healthfuel is a health tracking website that combines the interactiveness of other social media apps and the tracking capabilities of other fitness apps. Our goal is to help people track their fitness journey without the confusing layout of other trackers and without the distractions of some social media apps. However, we do agree that your fitness journey should not be a solitary one. This is why we allow users to connect to specifically like-minded individuals who share similar dedication to achieving their fitness goals. With Healthfuel, you are able to track your weight ( in lb or kgs), your calories, and the progress you've made by the month or week. The User will also be able to share their own recipes, exercises, and challenges with one another. With Healthfuel, you will have the ability to track your weight using either pounds(lbs.) or kilograms(kg), daily calorie intake, and progress on either a monthly or weekly basis. Sometimes, it can feel overwhelming or daunting at the prospect of making a long term commitment to your physical health.With Healthfuel, you have the opportunity to market to a wider audience by tapping into the growing health and fitness market. You can use Healthfuel as a platform to showcase your brand's commitment to health and wellness, while also engaging with your customers and building a community around your brand. To ease our users along, we encourage them to set short term “mini-goals” which act as milestones in order to capture the user's motivation and keep them disciplined. While you are focusing on these “mini-goals”, you may end up reaching your initial larger goal before you even realize it. HealthFuel is a great tool for anyone looking to take control of their health and fitness journey.

**Usability Test Plan**

Test Objectives:

We’ll be testing the ‘Upload’ and ‘Post Feed’ features of our app. There will be multiple feeds within the app, however this is the most important one as we anticipate it’ll be utilized the most by the user. This is the “Progress Pictures” feed. Based on our research, taking progress pictures while on a health journey is one of the most effective ways of keeping people motivated and disciplined in the gym.

We want this to be a very straightforward and easy feature to use so we aimed to have the least amount of steps for the user, while maintaining the best results we can get. Our goal with this test is to determine if there are any aspects of the function we may have overlooked that would need to be fixed or changed in a certain way.

Test Plan:

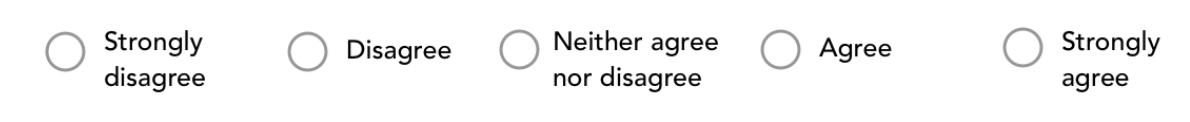
We will be using the XCode simulator of an iphone 13 pro to conduct this test. We have all our users’ information linked and saved on the Parse framework database and it’s installed on the app’s Workspace Podfile. We’ll be using the XCode simulator to run the app and have users log in as usual and go through the process of segueing into their photo library, scrolling through and finding the photo they’d like to upload, adding a caption if they want, and finally uploading the photo.

The task at hand is making sure uploads are made with ease and efficiency. Since this is an app meant to assist those who are trying to better themselves, our target audience is a quite large pool of users. For this reason, we’ve decided to diversify our test cases as much as possible; an upload feature like this one is very similar to those of social media apps so we anticipate that users who are active users of those apps should not have any issues uploading their content with ease. The majority of people who are frequent users of said social media apps range between teenagers and late 20’s; this coincides with the majority of people who workout being within that same age group. So now that our research has shown that the majority of our target audience is expected to have no issues with this feature, we’ll be focusing on the minority demographics as we’d like this app to be comfortable and easy to use for anyone trying to better themselves. The two main minorities within our target demographic are users over the age of 45, and users with a language barrier (this was found to be an issue with many health and fitness apps as everything is written out in a way where there are multiple steps needed to do most things, which would be very difficult for someone who isn’t familiar with the language). Our strides in eliminating this issue was making the process take the least amount of steps possible, and adding icons/photos to relatively every step so a user can be able to do it successfully without having to read anything.

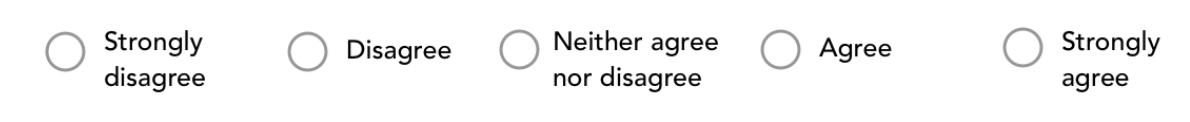
The completion criteria of this test will be successfully uploading a picture to the Progress Feed within the 60 seconds of starting the process.

Likert Questionnaire:

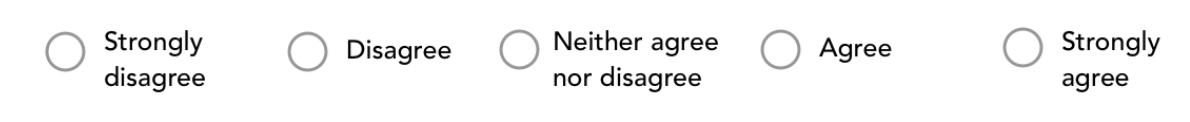
1. **Uploading my progress pictures was very easy and straightforward.**



1. **Uploading my progress pictures makes me feel more motivated and confident.**

****

1. **Uploading my progress pictures felt unnecessarily complicated and time consuming.**

****

Test URL:

<https://github.com/CEN4010-Group11/Group-11/tree/main/milestone%20%234>

**QA test plan**

Testing Product:**HealthFuel**

* **Feature being tested**: Uploading pictures through device’s photo album
* **Objective**:

The testing objective is to be able to properly login with test usernames and passwords, access the device’s photo album and upload to HealthFuel. After uploading, the users should be able to view the photo with their name ( author’s name) and caption underneath.

**List:**

* Testing the upload feature
* Display photo album to choose photos
* upload .png files
* view photos in the proper place
* Have photo’s caption and author saved with picture
* **Hardware/ Software Setup:**

| **Hardware** | Using simulator on PC (iphone 13) |
| --- | --- |
| **Software** | Interactive simulator ( via xcode) |

* **Test Cases:**
* Login with username/password
* Upload picture
* View picture with author and caption underneath

Browsers Used:

| Test case # | **Username/Password** | **Photos Uploaded** | **Caption/ Author** |
| --- | --- | --- | --- |
| #1 | **Username**: johnsmith  **Password**: password |  | **Caption:**”test”  **Author:** johnsmith |
| #2 | **Username**: johndoe  **Password**: password |  | **Caption:**N/A  **Author:** johndoe |
| #3 | **Username**: johnbrown  **Password**: password |  | **Caption:**”Testing123”  **Author:** johnbrown |

**Code review**

Note: The regular comments were made by the main coder and all the code review comments begin with “Code Review Comment:”

class CameraViewController: UIViewController, UIImagePickerControllerDelegate, UINavigationControllerDelegate {

@IBOutlet weak var imageView: UIImageView!

@IBOutlet weak var captionField: UITextField!

override func viewDidLoad() {

super.viewDidLoad()

// Do any additional setup after loading the view.

} /\* Code Review Comment: Function looks good. \*/

//button that would save and upload the desired image

@IBAction func onUploadButton(\_ sender: Any) {

let progressPost = PFObject(className: "progressPosts")

progressPost["caption"] = captionField.text! /\* Code Review Comment: I think some error handling to check if the user entered nothing for the caption should be added. \*/

progressPost["author"] = PFUser.current()!

//identifying image source, type, and destination in the database.

let imageData = imageView.image!.pngData()

let file = PFFileObject(name: "image.png", data: imageData!

progressPost["image"] = file

//automatically saving the image without the user having to manually add it again

progressPost.saveInBackground { (success, error) in

if success {

self.dismiss(animated: true, completion: nil)

print("saved!")

} else {

print("error!")

} } }

//added a 'tap gesture recognizer' to the image view so the user would be able to tap anywhere on the image view and it'll display their photo album to choose what image they'd like to upload.

@IBAction func onCameraButton(\_ sender: Any) {

let picker = UIImagePickerController()

picker.delegate = self

picker.allowsEditing = true

if UIImagePickerController.isSourceTypeAvailable(.camera) {

picker.sourceType = .camera

} else {

picker.sourceType = .photoLibrary

}

present(picker, animated: true, completion: nil)

} /\* Code Review Comment: Function looks good. \*/

func imagePickerController(\_ picker: UIImagePickerController, didFinishPickingMediaWithInfo info: [UIImagePickerController.InfoKey : Any]) {

let image = info[.editedImage] as! UIImage

let size = CGSize(width: 300, height: 300)

let scaledImage = image.af\_imageScaled(to: size)

imageView.image = scaledImage

dismiss(animated: true, completion: nil)

} /\* Code Review Comment: Function looks good. \*/

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

@IBOutlet weak var tableView: UITableView!

var progressPosts = [PFObject]()

override func viewDidLoad() {

super.viewDidLoad()

tableView.delegate = self

tableView.dataSource = self

// Do any additional setup after loading the view.

} /\* Code Review Comment: Function looks good. \*/

//creating the query that the progress posts will be saved to in the Parse database and identifying user/hidden keys.

override func viewDidAppear(\_ animated: Bool) {

super.viewDidAppear(animated)

let query = PFQuery(className: "progressPosts")

query.includeKey("author")

query.limit = 20

query.findObjectsInBackground { (progressPosts, error) in

if progressPosts != nil {

self.progressPosts = progressPosts!

self.tableView.reloadData()

}

}

} /\* Code Review Comment: Function looks good. \*/

//making it so the table automatically adds rows as more users add posts.

func tableView(\_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {

return progressPosts.count

} /\* Code Review Comment: Function looks good. \*/

//for more efficiency, we had the table reuse cells so instead of infinitely creating more and more cells, it'll just reuse any previously created cells that are not currently displayed on the screen; this will help prevent latency and overwhelming the backend server.

func tableView(\_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {

let cell = tableView.dequeueReusableCell(withIdentifier: "ProgressCell") as! ProgressCell

let progressPost = progressPosts[indexPath.row]

let user = progressPost["author"] as! PFUser

cell.usernameLabel.text = user.username

cell.captionLabel.text = progressPost["caption"] as! String

let imageFile = progressPost["image"] as! PFFileObject

let urlString = imageFile.url!

let url = URL(string: urlString)!

cell.progressphotoView.af.setImage(withURL: url)

return cell

} /\* Code Review Comment: Function looks good. \*/

} /\* Code Review Comment: Couldn't find any issues with this code, well done! \*/

**Best practice for security**

All user assets are saved and protected within the database. All passwords are encrypted in the Parse server and all user input is saved on the server under their account so there is no point in the process where any information would be mixed with, or exposed to another user’s info. All this can be seen in the “Milestone #4 parse database” video on our youtube.

Here’s the link: <https://youtu.be/cKpVXi_yJe8>

The links to the two youtube videos (this one, and one of the simulator running our app) can also be found at the bottom of this document and in the submission comments on canvas.

**Non-functional requirements**

1. Performance/Scalability

The results that are mainly needed to be returned would be the multi-user feeds. To prevent any latency, we will create a specific cell reload (ex: 20 cells at a time) with reusable cells so the software can call any cells out of view and reuse them to load more posts, instead of constantly creating new cells. **DONE**

1. Portability/Compatibility

We will be using XCode, mainly swift and ruby, to create an ios app that would be compatible with iphones. **DONE**

1. Usability

Part of our goal in making our users’ health journeys as successful as possible is creating a very straightforward and easy to use app. We do not wish to limit our users to only the tech-savvy so we are striving to limit as many steps as possible on then user’s end without jeopardizing app productivity. **DONE**

**Youtube Links**

App Simulator:

* <https://youtube.com/shorts/2E2eOT2lmt0>

Parse Database:

* <https://youtu.be/cKpVXi_yJe8>