# Milestone 4

Project Title: HeapOverload
CSC648/848-01
Team 4
Adam - Team Lead, Bishow - Scrum Master, Briana - Front End Lead,
Jasmeet - Front End Lead, Ming - Backend Lead, Sakar - Github Master
11/27/2022

## Product Summary

Name: HeapOverload

Functions:

Post System: Create Post, Delete Post, Edit Post, View Post

Account System: Login, Register

Tag System: Navigation/Exploration of Posts and other Tags

Unique Feature: Tag System, our goal is to make our tag system the main means of navigating for our application. It can be used to explore and find new posts that you might be interested in.

Product URL: http://heapoverload.com

## QA Testing

Issues: We had some delivery issues for this milestone, we got a bit behind schedule for getting tasks completed and this affected our timeline on creating tests. There was some time that we spent just trying to get things to work as well as making the backend hosted in the cloud for easy access to begin testing. With these issues we struggled to get the backend to a point that we were able to really start testing all the features or even at all. However we were able to accomplish some integration tests that will be below.

Integration Testing:

Test case 1:

Test case description: To create a post

Date tested: 11/27/2022

Test scenario: Create post

Prerequisites: Access to Chrome Browser

Test data: “title”: “Fifa”, “author”: “Messi”, “content”: “I love soccer.”, “tag”: [“fifa”, “soccer”]
Expected result: True
Actual result: True

Test data 2: “title”: “Fifa”, “author”: “Messi”, “content”: “I love soccer”
Expected result: False
Actual result: False

Test case 2:
Test case description: To register for an account
Date tested: 11/27/2022
Test scenario: register account
Prerequisites: Access to Chrome Browser
Test data: “email”: “briana@gmail.com”, “password”: “sakar123”, “username”: “sakar1”
Expected result: True
Actual result: True

Test data 2: “email”: “briana@gmail.com”, “password”: “sakar123”, “username”: “sakar1”
Expected result: False
Actual result: False

Test case 3:
Test case description: To login to the website
Date tested: 11/27/2022
Test scenario: login account
Prerequisites: Access to Chrome Browser
Test data: “email”: “sakar@gmail.com”, “password”: “sakar123”
Expected result: True
Actual result: True

Test data 2: “email”: “sakar@gmail.com”
Expected result: False
Actual result: False

With this we were able to test the integration of the account system as well as the post creation APIs. The other functions are nearly working and will be tested and functional for the delivery date.

## Code Review

For this we layed out some stylistic choices that we all agreed on in the beginning:
There are the guidelines and stylistic choices that we are going to follow the best we can when writing code for this project. If you have any issue with these styles please let me know and we can vote to change or make an addendum.

## Variables

For mutable variables please use descriptive names as well as camel case starting with a lowercase letter for any mutable variable (examples below I am trying to use generic pseudocode terms):

```
var mutabaleVariableName
var likeCounter
```

For immutable variables please use descriptive names as well as all the letters capitalized and '_' to represent a space:

```
var IMMUTABLE_VARIABLE_NAME
var WEBSITE_URL
```

## Functions

Functions should have a descriptive and meaningful title and camel case starting with a lowercase letter. Any input values should also have a meaningful name.

```
func divideTwoNumbers(var dividen, var divisor) {
    return dividend/divisor
}
```

## Classes

Classes should have a meaningful name and be camel cased, starting with a capital letter. Member variables and functions should also have meaningful names.

```
class Fruit {
            var kindOfFruit
            func getKindOfFruit() {return kindOfFruit}
      }
```

## Comments

Comments should be in a block form before a class or function. My hope is that your code is readable enough that you shouldn't need to comment on each line.

Class comments should be descriptive of what the class is used for as well as what member variables and functions that it has:

```
/*
*  The Fruit class is used to classify fruits.
*  Member Variables:
*       kindOfFruit: This will hold a string that denotes the type of fruit
*  Member Functions:
*       getKindOfFruit(): This function will return the value stored in kindOfFruit
*/
class Fruit {
      var kindOfFruit
      func getKindOfFruit() {return kindOfFruit}
}
```

Function comments should be more descriptive than the name and state what each input is and what, if anything will be returned or what is done by calling this function:

```
/*
*  This function takes 2 numbers as input and return the quotient
*/
func divideTwoNumbers(var dividen, var divisor) {
      return dividend/divisor
}
```

## Indentation

This is key to very organized and readable code. Code that is contained inside brackets should be indented depending on how deep the scope is.

```
func exampleOfIndentation(){
      if (true){
            while (true){
```

```
                        if (true){
                                print('Test')
                        }
                }
        }
}
```

For this, we do not currently have github pull requests as Adam has personally handled combining files together and made sure that things are working together properly.