Software Requirement Document
NutriTrack Nutrient Tracking Application

Onat Bas
Bogazici University
Computer Engineering Department
onatbas   gmail.com

December 19, 2016

Contents

1 Introduction **3**

1.1 Project Description 3

2 Requirements **4**

3 Design Document **7**

3.1 High Level Architecture 7

3.2 Behavioural and Functional Design 7

3.2.1 Sign-up Process 9

3.2.2 Sign-in Process 11

3.2.3 Food Search Process 12

3.2.4 Activity Search and Tracking Process 13

3.2.5 Food Tracking Process 14

3.3 High-Level View and User Interface Design 15

3.3.1 Initial Design 15

3.3.2 Main View 16

3.3.3 Screen 2 17

3.3.4 Screen 3 17

3.3.5 Screen 4 18

4 Plan **18**

4.1 Project Development Plan 18

5 Deployment **19**

5.1 Description 19

5.2 Deployment 19

5.2.1 Production 20

5.2.2 Admin Account 20

5.2.3 Creating Database 20

5.2.4 Production 21

1 Introduction

1.1 Project Description

NutriTrack is designed as a Web application with which users can track their nutrition values by providing what they eat and their activity information by providing their daily activities.

This document is created to serve both as a design document as well as a detailed description. Requirements and technical information will also be provided.

The recommended way of reading this document is to have a digital copy available nearby so you may see the referenes while you're reading the document.

2 Requirements

1. Users adapt a healthier lifestyle
   (a) Functional Requirements
      i. Users should sign-up to
         A. Personal information is collected during user sign up
         B. Users that have already signed up are able to login
      ii. User Personal Information Input
         A. User inputs static data (Gender, Date of Birth)
         B. User inputs variable data (Height, Weight, Notes)
         C. Variable data is updateable
         D. Variable personal data is kept as history
      iii. User Food Consumption Input
         A. User creates menus and re-uses these menus
         B. User selects the group of food item he is searching
         C. A textbox is used to get input from user
         D. USDA database is used to query food that the user inputs
         E. The unit of the selected food is queried from the USDA database
         F. A drop down menu of food items that match users search are presented and user selects from this list
         G. User inputs the amount of given food where the interface prints out the unit
         H. User inputs food consumption data for a given date
         I. User is able to select a date that is different than current date
      iv. User Physical Activity Input
         A. http://www.nutristrategy.com/activitylist4.htm is used to query physical activity
         B. Physical activity is grouped as daily activity and sports
         C. User inputs the duration of given physical activity
         D. Minutes are always used as unit of physical activity duration

v. Insight on User Fitness
   A. Software gives insight on calorie comparison
      - Total calorie intake from the food user has consumed is known
      - Calorie burn of users physical activity is known
      - Overall calorie intake and output is compared
   B. Software gives insight on nutrition consumption
      - Nutrition intake from the food user has consumed is known
      - Daily nutrition intake recommendations for users profile is known
      - Overall nutrition intake and recommendation is compared
   C. Software gives insight on Body Mass Index
      - Weight and Height of user is known
      - BMI calculation is done
   D. Nutrition and Calorie Analysis Over Time
      - Past data on nutrition input, calorie input and output are known
      - User selects an interval for analysis
      - Software provides intake and output comparison over the selected interval

(b) Non-functional Requirements

  i. User continuity
    A. The user returns to the application
    B. The user inputs most of his daily activity and food consumption
  ii. User Experience
    A. The user easily understands nutrition and calorie analysis provided by the software
    B. The user easily inputs food consumption and daily activity data
  iii. Software Technology
    A. The system is developed using Tomcat/Java
    B. MySQL database is used
    C. USDA API is consumed
  iv. Security
    A. Unique attribute of a user is his email address
    B. Every email address is associated with a password
    C. Data about a user is only visible if email and password data are present

(c) System Requirements

  i. System must be python2.7 compliant.
  ii. System must operate on ubuntu14.06 32bit or higher distribution.
  iii. System must be run with Django 1.10.3, explicitly.
  iv. System requires a database with SQL standard compliance.
  v. Network must be publicly available for given network port.

3 Design Document

3.1 High Level Architecture

In the application, client-server architecture is being used.

The server side is built with Django version 1.10.3. Due to frequent changes in the django environment, sticking to this version is advised for further development. Django API along with ”networks” library in python is responsible for making calls to USDA service. The USDA service has a API limit. To leverage that, the system includes an internal cache database that's on the same level with the USDA application which caches the queries, and the results of the queries in the system so that when a query is remade, it will be ready to use in our database. This behaviour will make better use when users are searching for food with few letters only.

The client side is built with HTML, CSS and JavaScript. In the development, some frequently used components are used, Namely Bootstrap, bootstrap.js and jquery. Their use has been mostly to validate user inputs on forms like registration, search etc.

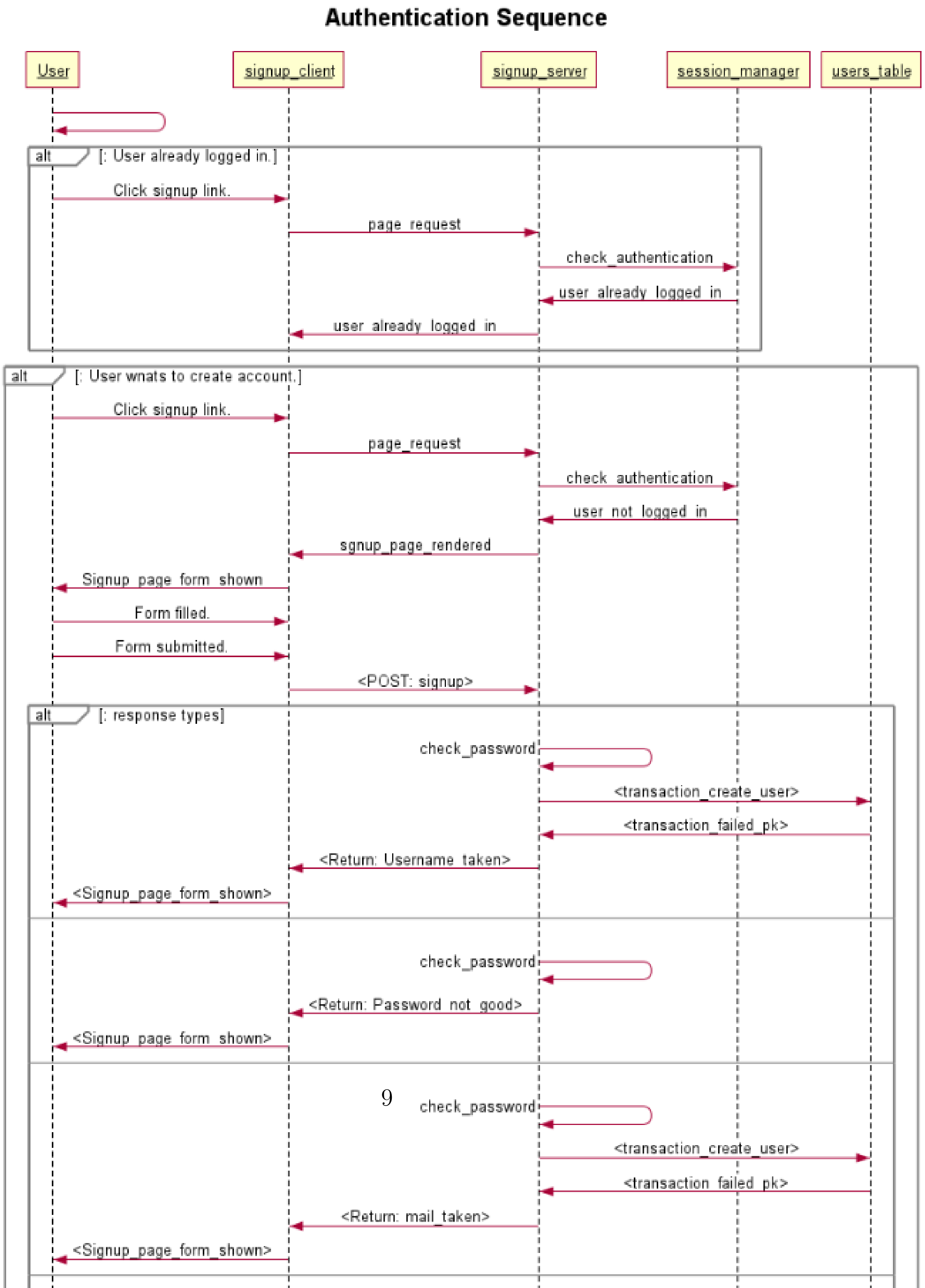
As a database, there's no restriction, however, for ease of use SQLite is recommended and is used in initial deployment as Django handles it.

In this system's design, cookies and browser caches are not used, not depended on.

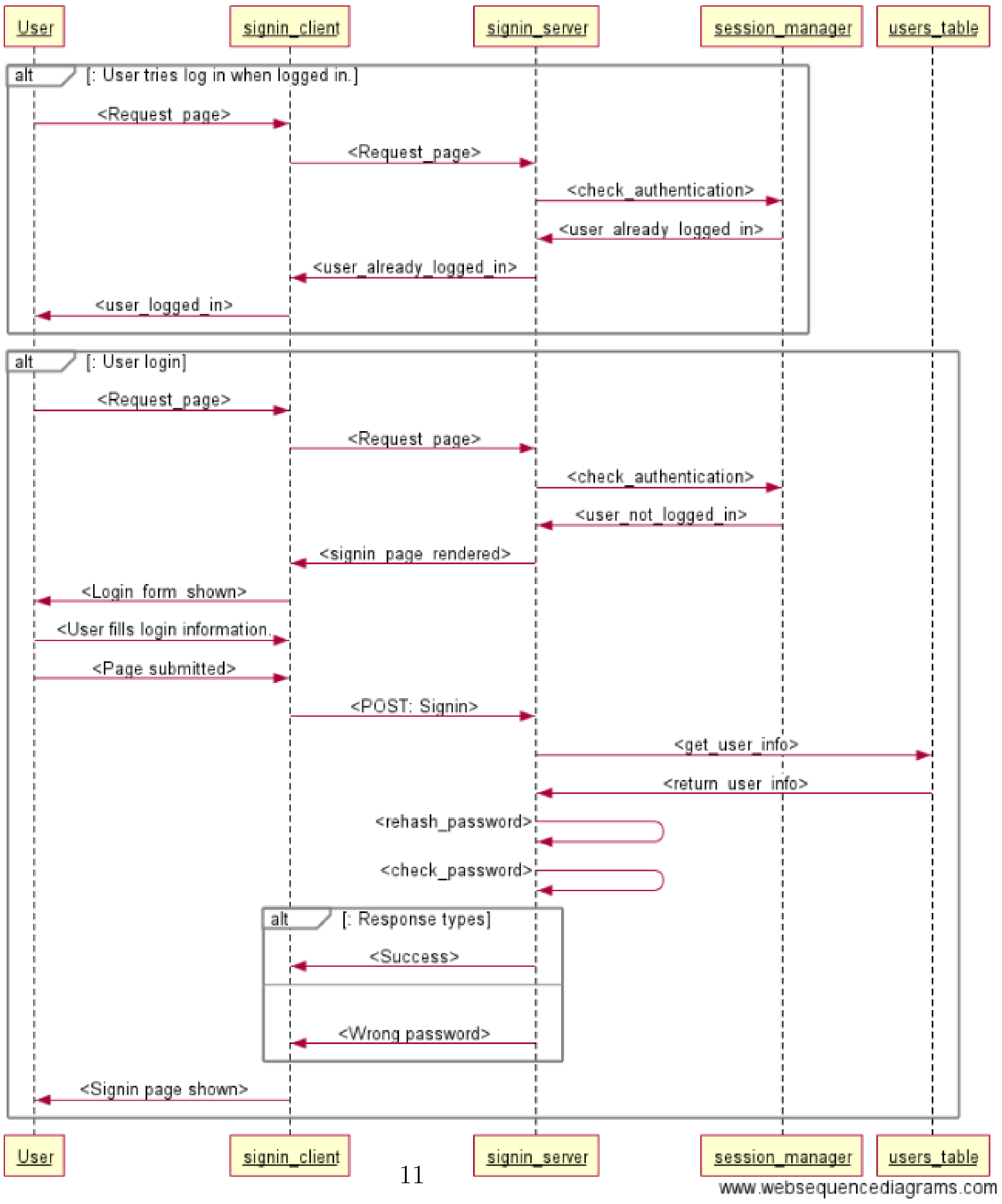
3.2 Behavioural and Functional Design

In this section, processes will be explained with sequence diagrams.

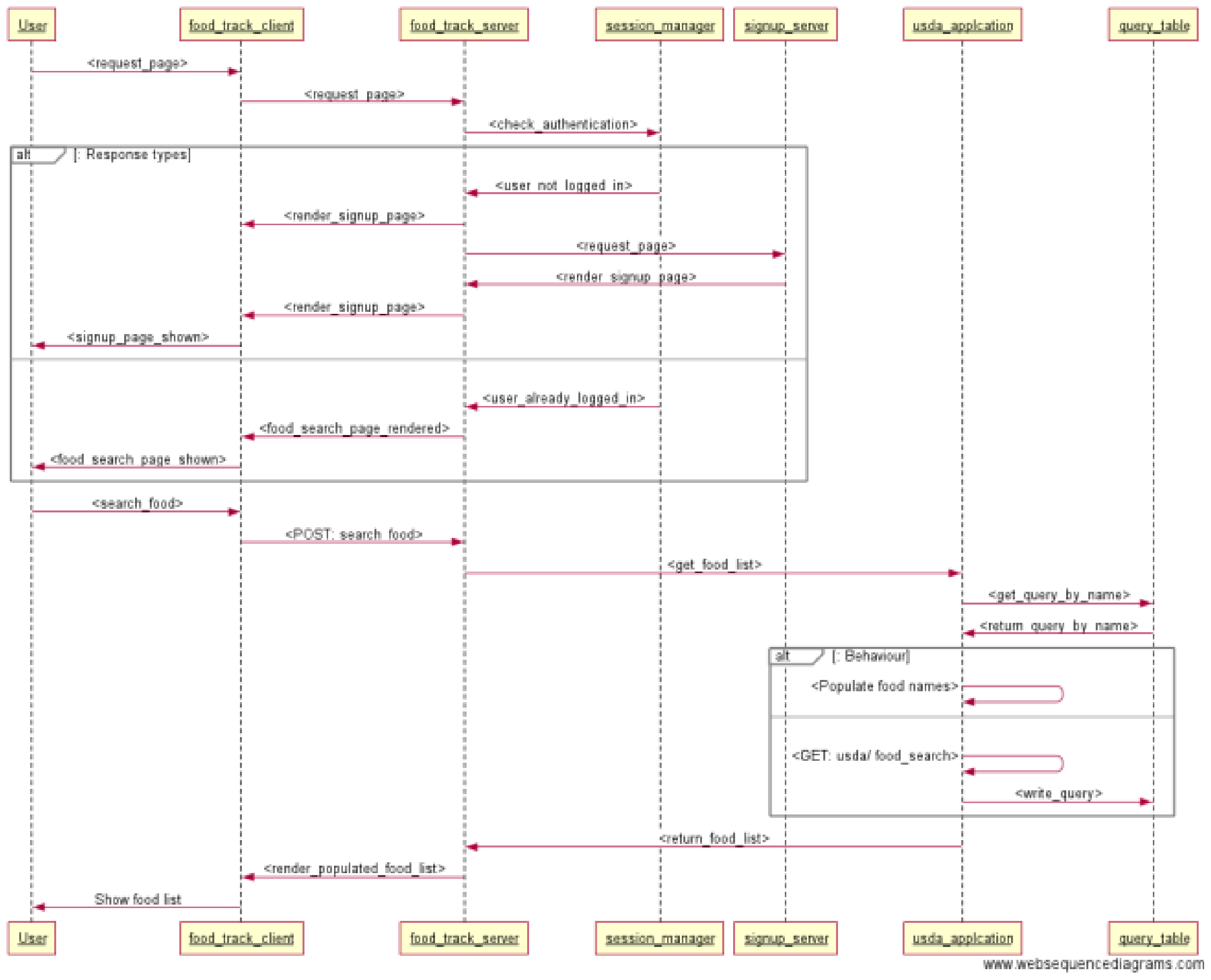
3.2.1 Sign-up Process



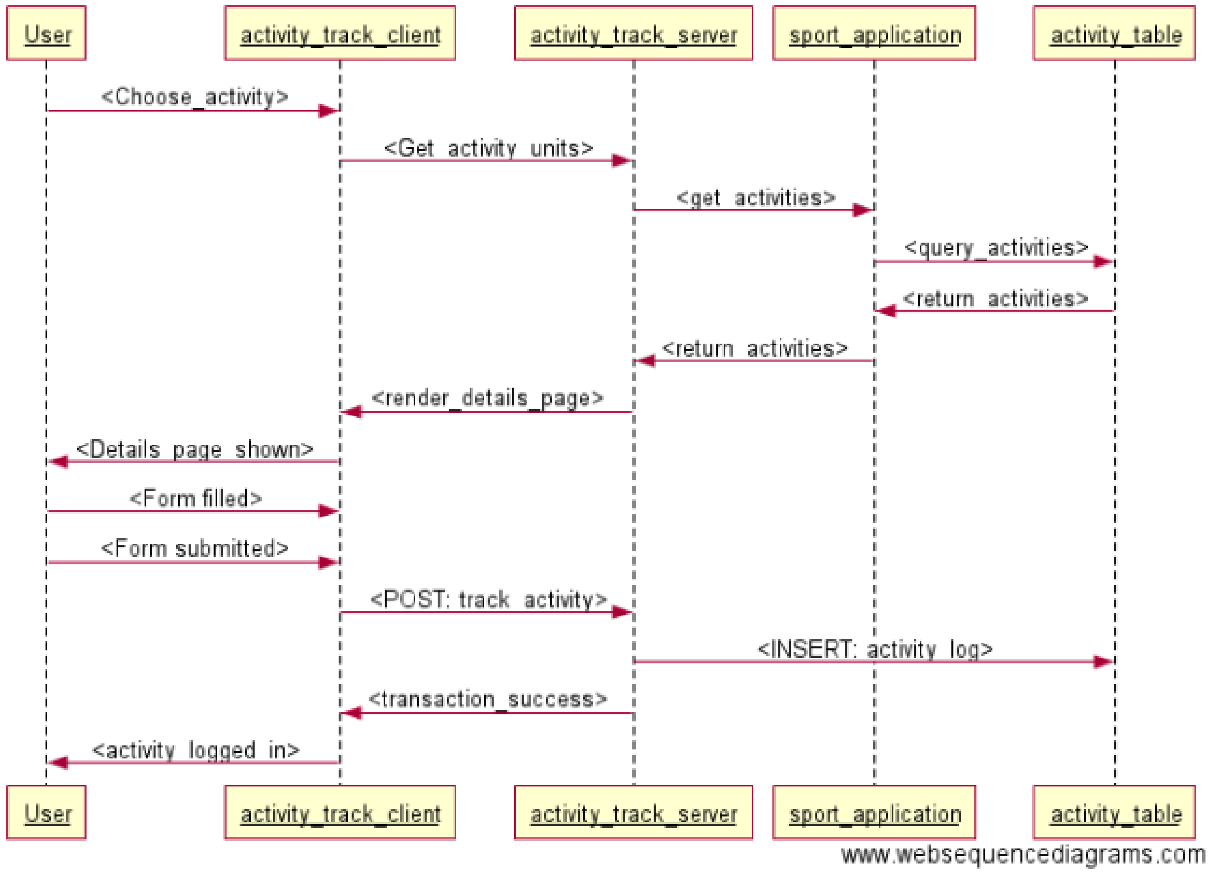
3.2.2 Sign-in Process



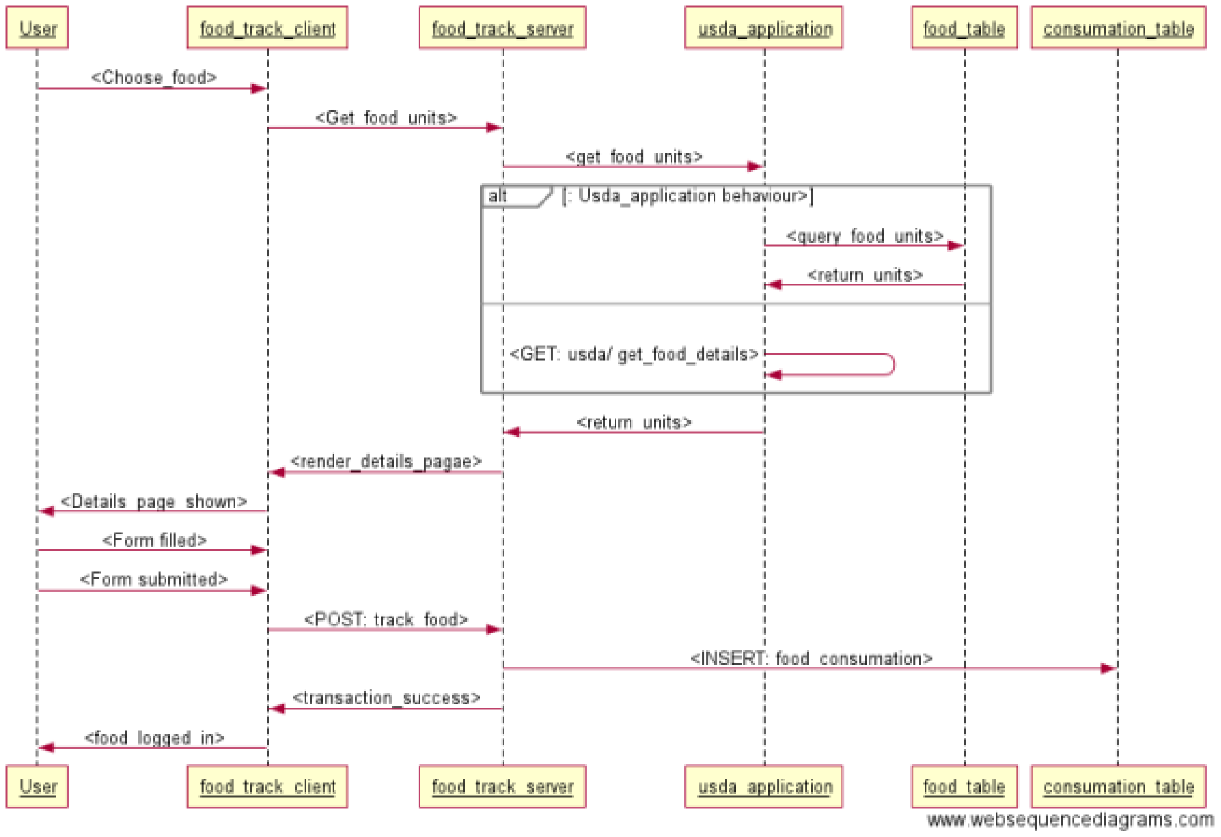
3.2.3 Food Search Process



3.2.4 Activity Search and Tracking Process



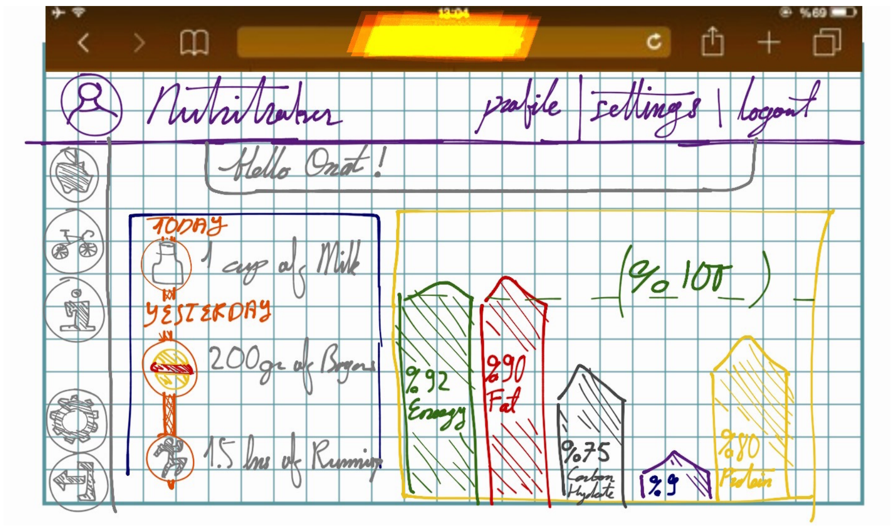
3.2.5 Food Tracking Process



3.3 High-Level View and User Interface Design

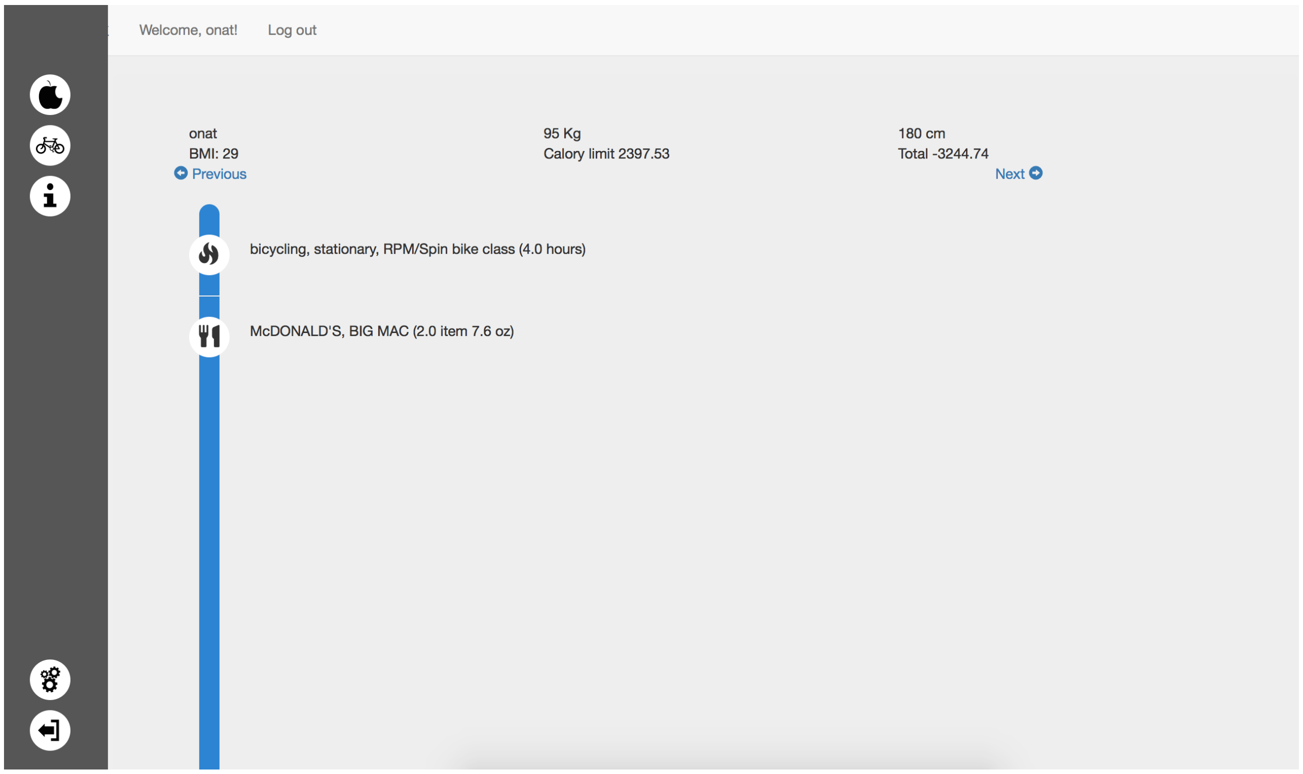
In this section, the design will be explained in iterative steps. First, draft will give an idea of how the webpage is ideally designed to look like. This first sketch is drawn by-hand on purpose because, design initially doesn't really rely on the tools.

3.3.1 Initial Design



This design has been designed in HTML/CSS for a technical view. Here's the initial mockup. The initial mockup doesn't include the graph charts.

3.3.2 Main View



onat
BMI: 29

95 Kg
Calory limit 2397.53

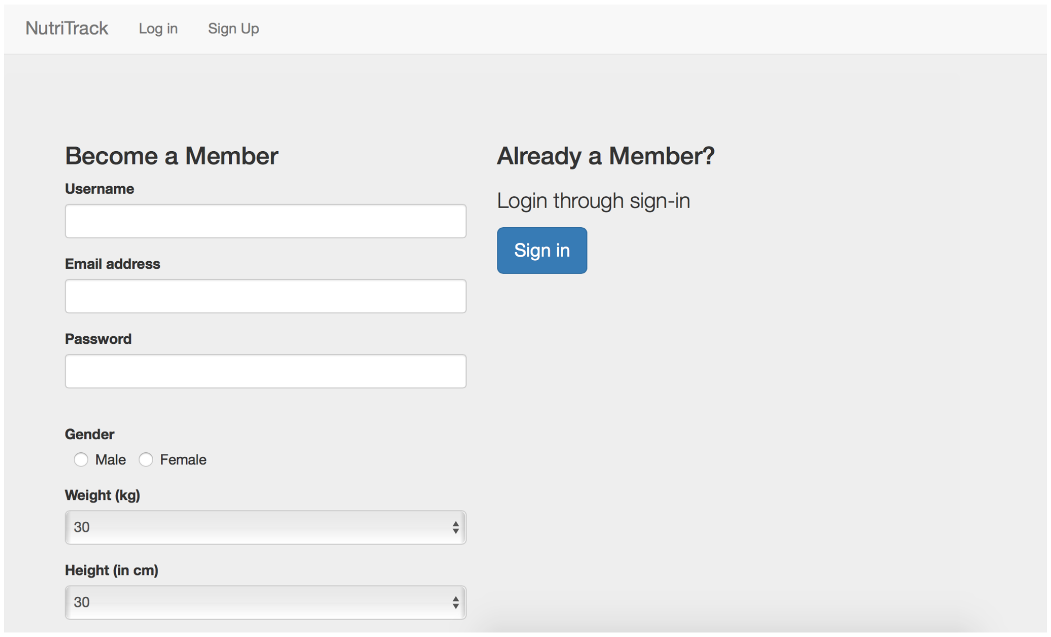
180 cm
Total -3244.74

bicycling, stationary, RPM/Spin bike class (4.0 hours)

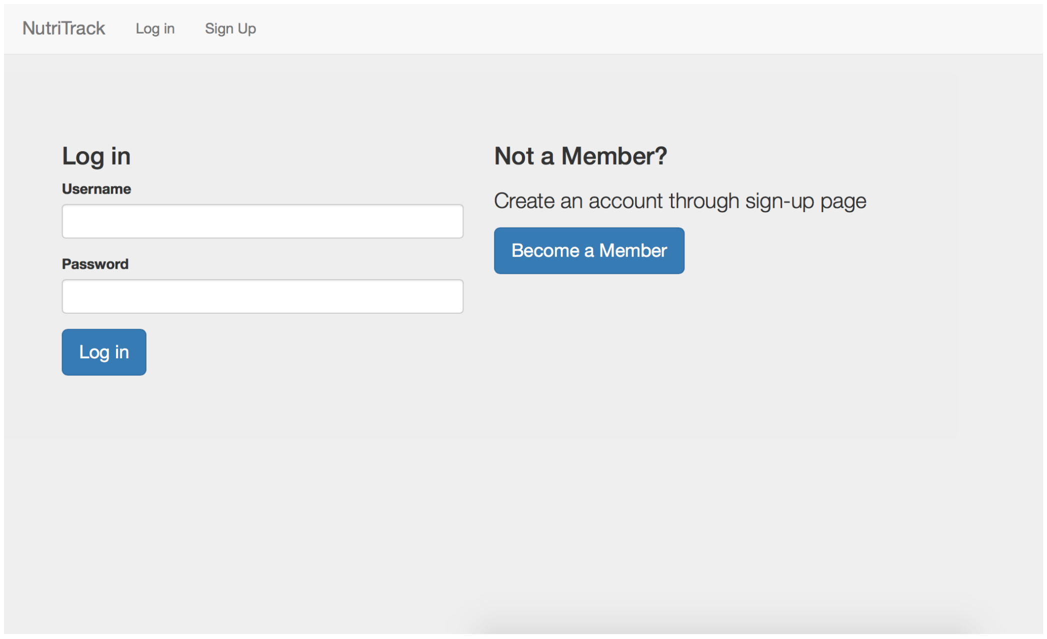
McDONALD'S, BIG MAC (2.0 item 7.6 oz)

This view is implemented with Bootstrap elements along with custom CSS scripts. With this general concept in mind, further mockups with similar styles are created.

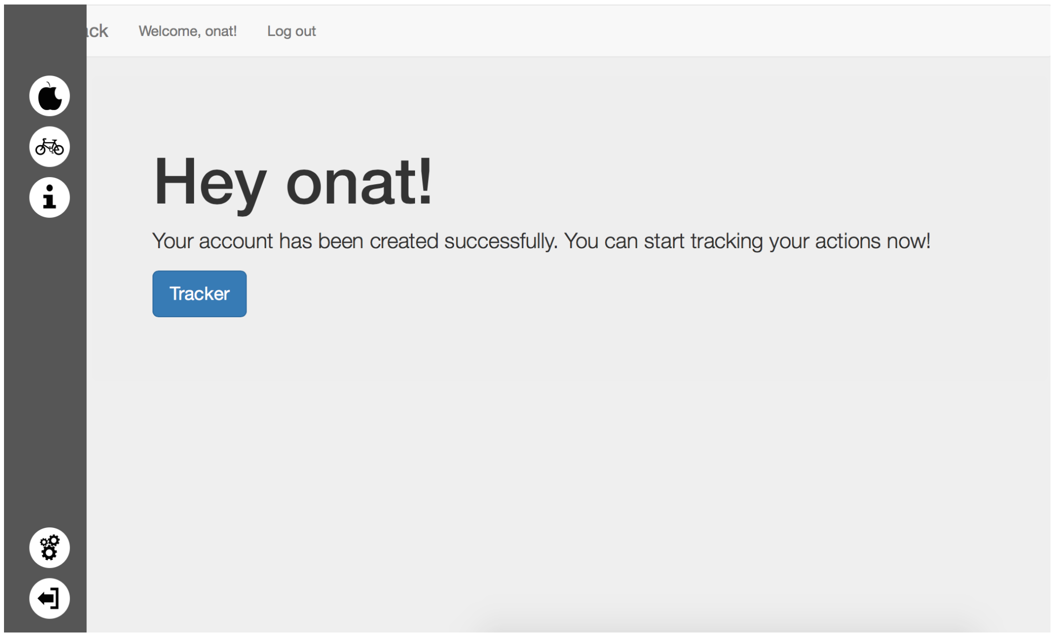
3.3.3 Screen 2



3.3.4 Screen 3



3.3.5 Screen 4



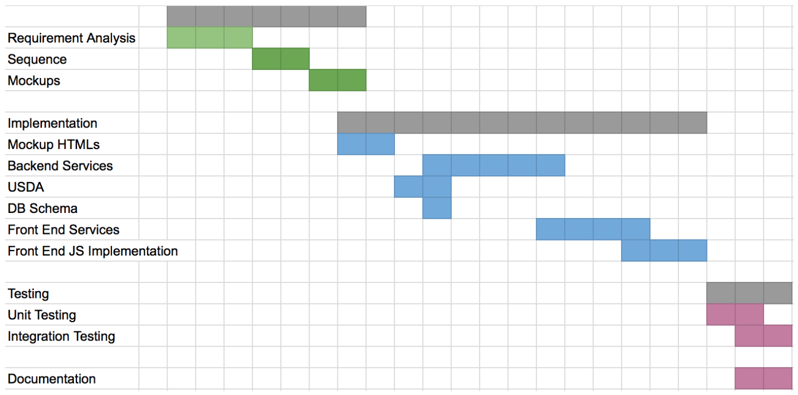
Hey onat!

Your account has been created successfully. You can start tracking your actions now!

4 Plan

4.1 Project Development Plan

The initial time estimate were made on estimates on three major sprints. You my view the estimates in the following figure. Each colored cell represents a 5-day period.



5 Deployment

5.1 Description

In this section, the steps which must be taken to deploy the application into production will be explained. To start with, the production environment must have these dependencies already installed:

Table 1: My caption

	Type	Name	Version
1	OS	Ubuntu	14.04+, 32bit+
2	Programming Language	Python	2.7
3	Dependency management	Pip	8.1.1+
4	Version control	Git	1.0+
5	SSH Agent	OpenSSH	1.0+

These dependencies are required to be in the production environment, and their installation processes are not covered in this document. However, many of these packages are usually included in standard linux distributions. If Ubuntu 16.04 is chosen as target OS, dependencies 2,3,4,5 will be available in the system.

Here are the dependencies of which installation processes are covered in this document

Table 2: My caption

	Type	Name	Version
1	Web Framework	Django	1.10.3
2	Network library	requests	Most recent

5.2 Deployment

The project is hosted at Github. Please clone the repository via Git:

```
$ git clone https://github.com/onatbas/SWE573
$ cd SWE573
```

A dependency.txt file is placed under the repository. This will install "requests" library for you. Type in this:

```
$ sudo pip freeze > dependencies.txt
```

You'll also need to install django manually. To ensure you have the correct version, please check;

```
$ python -v
```

gives you 2.7.
If Python version is correct, please install django with the correct version tag, like this:

```
$ pip install django==1.10.3
```

Based on your credentials, you might need to execute this on sudo mode.

5.2.1 Production

The current design of the application is based on django's ability to serve the application. This is done by "runserver" command. Before explaining this, some setup is required. This is to ensure;

1. A database is ready to use.
2. An admin account is created.

5.2.2 Admin Account

To create the admin account, simply type this in the terminal

```
$ python manage.py createsuperuser
```

and type in the required fields.

5.2.3 Creating Database

Django creates the database as long as the project has ORM information. In Nutritrack, these ORM files or migrations are also versioned with the project, so django has everything to create the tables, schemas etc. For quick SQLite setup, use this command.

```
$ python manage.py migrate
```

5.2.4 Production

After previous steps, NutriTrack is ready to go live production. Simply, type this in:

```
$ python manage.py runserver
```

If you're using SSH connection to your server instance, you want to be able to exit the command line while the application is running, to do this, screen command may be used.

```
$ screen
// Press enter
$ python manage.py runserver
// CTRL + A + D
```

screen is a standard tool that comes with linux dsitributions and is reliable to execute commands on other terminal instances while another process is running.

You may connect to the same screen instance with this call.

```
$ screen -r
```

If the instance has more than one screen instances running at a given time, this command will list the screen instances instead of connecting to a random one. From the list, id can be read and connection can be made based on the id of the screen such as this:

```
$ screen -r 27830
```

To kill all the screen instances, simply type this:

```
$ killall screen
```