

Documents Package

docs on karmacircle web app

Nur Ertem Unden | SWE 573 | 19 December 2016

Table of Contents

[All the files, docs are under the cloud repo shared with edit permissions: 2](#_Toc469966640)

[Overview 2](#_Toc469966641)

[Project Plan Outline 3](#_Toc469966642)

[Requirements Document 5](#_Toc469966643)

[Software Design Document 7](#_Toc469966644)

[Architecture Component Diagram 7](#_Toc469966645)

[Django simple Model View Controller Structure 8](#_Toc469966646)

[Use case diagram 9](#_Toc469966647)

[Entity Relationship Diagram with Crow Feet Notation 11](#_Toc469966648)

[User Activity Sequential Diagram 12](#_Toc469966649)

[Django generic Class Diagram 13](#_Toc469966650)

[Mockup diagrams 13](#_Toc469966651)

[System Specifications Document 21](#_Toc469966652)

[System Specifications Summary: 21](#_Toc469966653)

[Making the app live under already running EC2 instance with the only available Virtual Private Cloud VPC association configured 22](#_Toc469966654)

[Test Documents 24](#_Toc469966655)

[Simple UX and data storage test 24](#_Toc469966656)

# All the files, docs are under the cloud repo shared with edit permissions:

<https://1drv.ms/f/s!AjsBLtiXHuuRgrQSwE4OVFnDDqga1Q>

# Overview

Karma Circle web application was intended to be a responsive web app with cross platform and cross device capabilities. The common aim of the web app is to be able to offer an easy platform to possible users to track and monitor their daily health behavior and turn it into an attitude. The behaviors to be tracked are grouped into 2 main categories under physical and emotional. Physical aspect includes: food consumption and activities, whereas emotional tracking involves mood tracking along with these.

Although the intentions and outline was very clear, it was a big challenge for the author to apply these intentions into practice due to lack of competency and experience in the field of software development as a practice.

This is the first hands on full web stack development experience of the author and it was an immense multi-dimensional challenge in an unchartered territory foreign to the author.

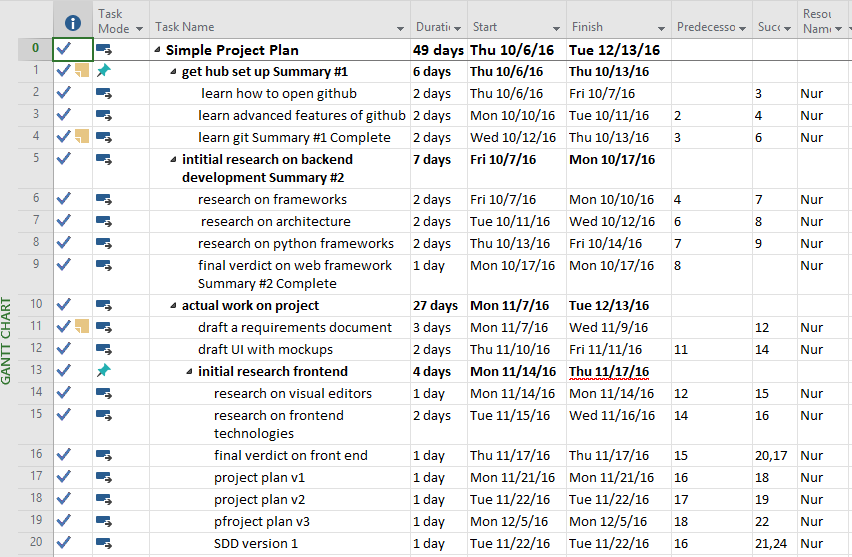
Although the author is up to take new challenges, the vast knowledge to be consumed is overwhelming. The author was guided to take the challenge offered as a medium for learning also, rather than just focusing on the deliverables.

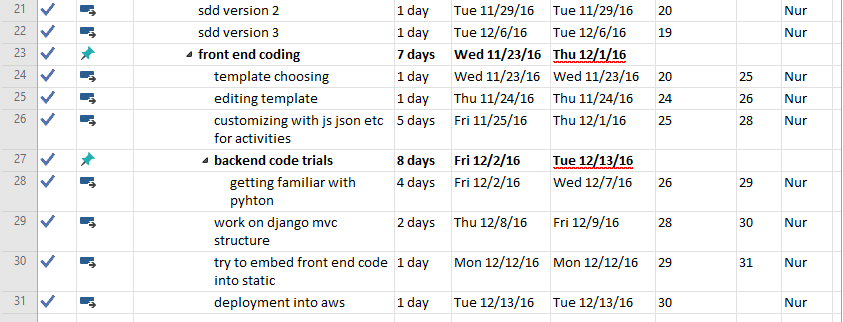
If we revisit the main idea of the web app; it was to offer a platform for human beings to track their behavior, turn them into habits and then into attitudes. The main idea of people being creatures of habit was expressed over and over again in the classroom while trying to exemplify the situation. The overall process of establishing a persistent behavior requires repetitions and experience. It is also suggested that one needs 10,000 hours to master a skill. In line with these facts, and after the project plan was outlined by the author; it was evident that the tasks at hand was far from achieving with the existing skills and experience. The aim was to turn things into best effort basis, just like trying to exercise everyday with only 10 minutes and then eating healthy at least once a day.

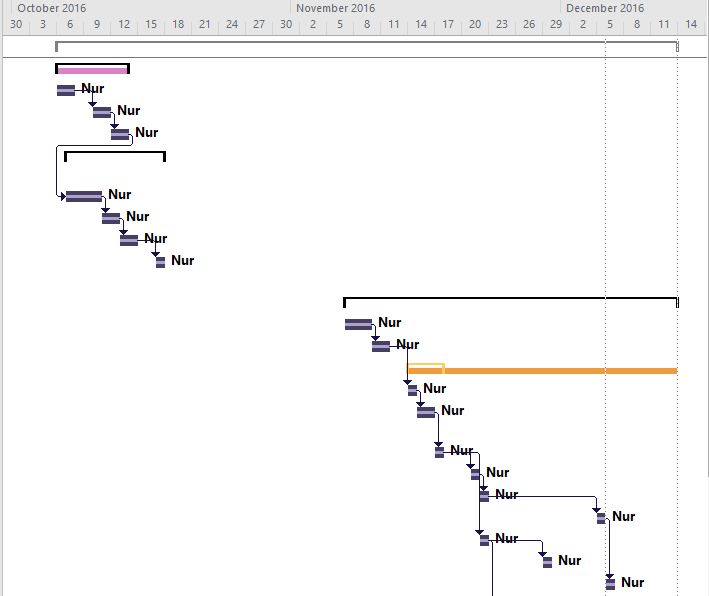
In the bigger picture of things, Karma Circle web app development offers a “karmatic” stage for the author to develop herself in the field of software development.

# Project Plan Outline

Each workday in the below plan corresponds to 3 hours of working time between 11:00 – 14:00 on week days only.









# Requirements Document

**REQUIREMENTS DOCUMENT FOR KARMA CIRCLE HEALTH TRACKING WEB APP**

The web app is simply designed to track calories gained and calories spent during the day. The user will be able track and monitor his progress daily, monthly and annually. The app will provide guidance, more like a reference point, and help the user to observe his behavior in order to try to have a healthy outlook. The application also incorporates a daily emotional rating scale. The mood tracking ability tries to bring a comprehensive view to the overall health of the individual, and distinguishes the application by design from its counterparts.

1. **Functional Requirements**
2. **Account creation basic requirements**
3. **User shall be able to sign up by filling out all the available form fields defined by the web application** 
   * 1. User must be able to fill out the name field with letters, last name field with letters, e-mail field in the format of [name@domain.com](mailto:name@domain.com) and password with any character combination. There are no specific restrictions limiting the entry of the user
     2. User shall be able to login to the account using the e-mail address and password provided to the system during the sign up process as the unique ID.
     3. User shall be able to enter/edit the account details under the account created.
     4. User must be able to enter personal details in order to be able to use the system like; birth date in dd.mm.yy format, sex as female or male, height in centimeters and weight in kilograms.
4. Nutrition tracking basic requirements (calories in)
   1. User must be able to pick a date to be able to track the activities in dd.mm.yy format.
   2. User shall be able to add food to the account he logged in with. User must be able to search food by writing down the name, choose the corresponding unit offered by drop down, and to add it to the account.
   3. After adding the food to the account, user shall be able to display the generic nutrients list associated with the food added.
   4. User should be able to see the last food added to his account visible at the top of the search box where he searches for new food to add.
5. Physical activity tracking requirements (calories out)
   1. User must be able to pick a date to be able to track the activities in dd.mm.yy format.
   2. User shall provide physical activity name in plain language and choose from the list and add it to the account by providing name, duration in minutes.
   3. User should be able to see all activities listed and easily pick up from available list offered.
   4. User shall display the calories spent for that specific activity.
6. Mood tracking requirements
   1. User must be able to pick a date to be able to track the activities in dd.mm.yy format.
   2. User shall provide the mood rating by selecting one of the five available ratings offered and save it to his account.
7. Guidance providing requirements
   * 1. Basal Metabolic Rate in calories will be utilized to provide guidance to the user by providing it as a denominator to calories in and calories out in the home page which displays the summary of the day in terms of calories.
8. History of calories in and calories out
   1. User must be able to pick a date to be able to view the activities in dd.mm.yy format.
   2. User shall be able display how many calories in as one line graph and calories out as another line a single graph with dates displayed in x axis and calories displayed in y axis.
9. **Non-functional requirements**
10. The system shall be able to be expandable with new users and features if needed.
11. The system should be self-explanatory at a glance with very simple UI offering very little graphic details.
12. The system shall have responsive user interface so that cross platform and cross browser adaptability is offered to the user.
13. The system will utilize open source technologies, tools, and offer free usage.

# Software Design Document

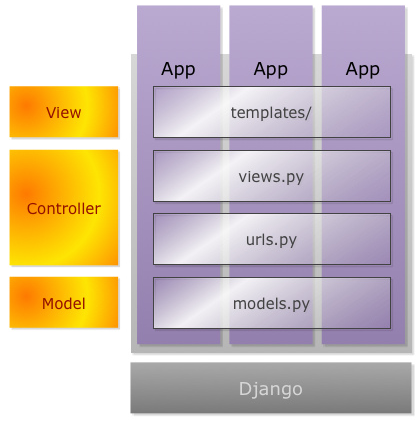
SDD will provide mostly a visual representation of the Karma Circle web app. First we explain the high level components of the architecture as below:

### Architecture Component Diagram



The web app is a classic example of the client server application along with very strong features of Model View Controller components which are inherently built in Django framework as a structure for ease of use. Also the default sqlite database provided by Django is used to store all user data.

### Django simple Model View Controller Structure



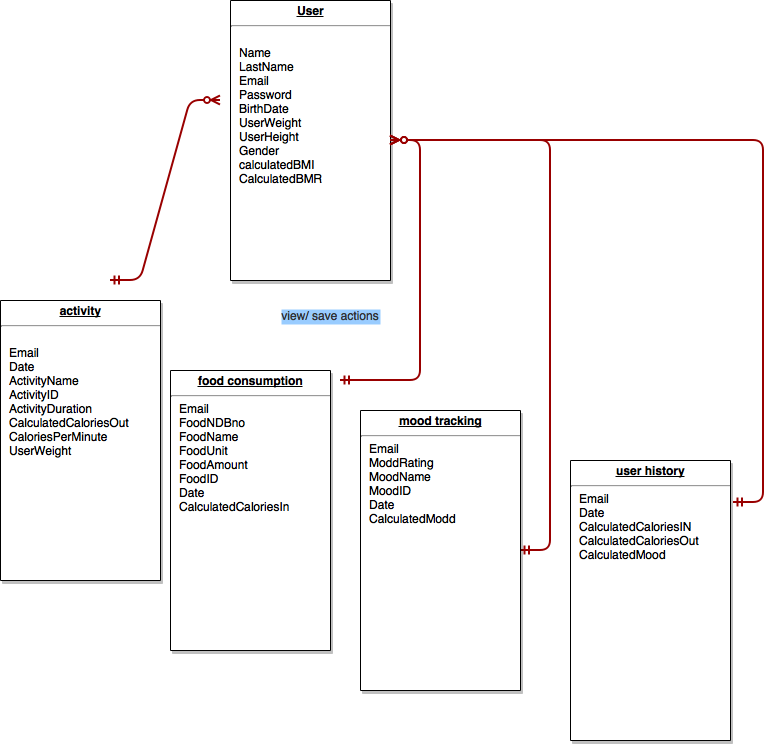
Probable user of the web app might want to see the below features “use cases” offered to him.

### Use case diagram



Now that we understand the overall picture and the needs of a probable user, we can start working on details of the web system. ERD will be presented below to give a general idea of the entities with attributes in the system and their basic relationships. The crow foot notation is preferred over Chen’s ERD system because of the ability to be able to list the attributes as a list. The below notation also removes the need an extra data model diagram to be offered because it gives an understanding of the attributes to be used, only types are missing, which is pretty obvious to the developer in most cases.

### Entity Relationship Diagram with Crow Feet Notation



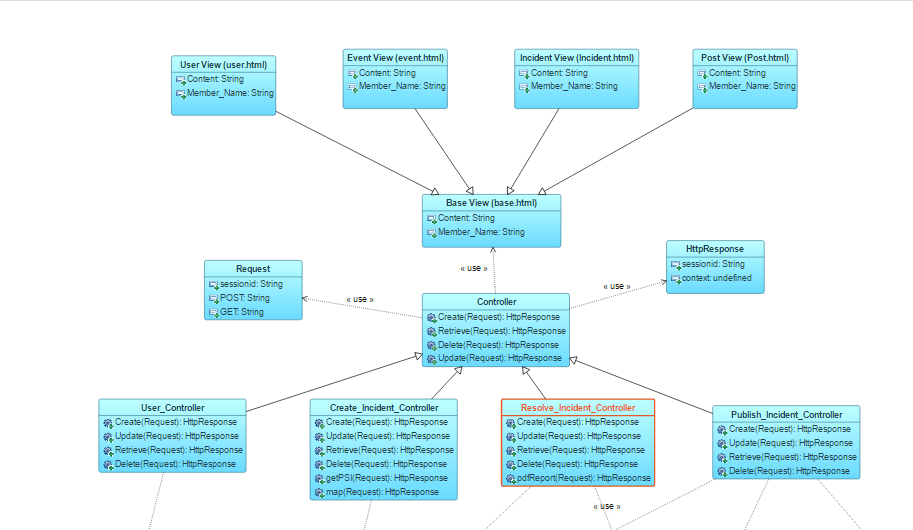
### User Activity Sequential Diagram

The newbie developer was only able to get the full activity schema working including front and back end.



### 

### Django generic Class Diagram



In order to understand the web app in detail mockups help tremendously

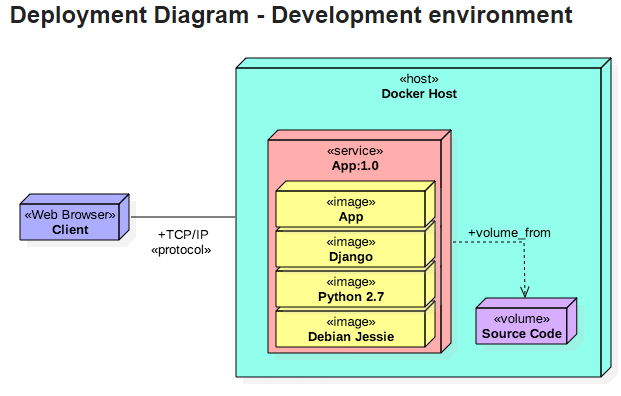
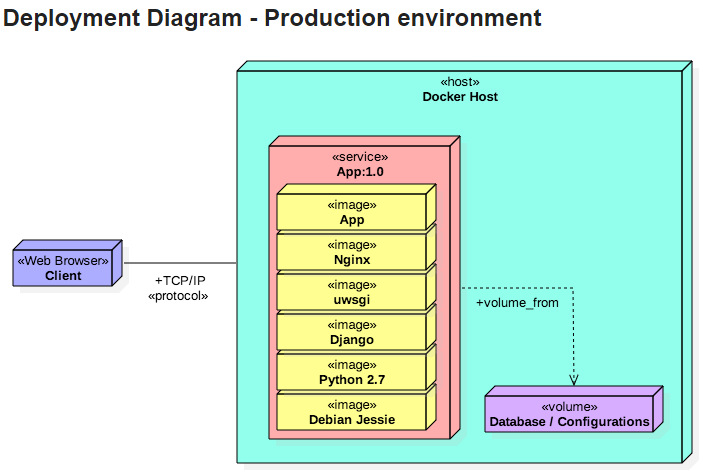
### Mockup diagrams





# System Specifications Document

The system is already deployed under Amazon Web Server with a static IP and port as <http://52.212.251.22:88/>. There seems to be a glitch with the Putty application and the structure of the production server. The web app runs without any errors under the deployment server when connected with putty. Here is a visual to explain the situation:

The developer hopes to upgrade to production with the below stated method:

### System Specifications Summary:

* Python3.5 as the OO programming language
* Django as the web framework
* Django-Rest
* Django requests
* SQLite as the default database storage
* 64-bit Ubuntu Linuxfree tier in AWS with Apache2 web server

### Making the app live under already running EC2 instance with the only available Virtual Private Cloud VPC association configured

The AWS is far from self -explanatory and documentation offered does not follow full scenario, it is just bits and pieces sewn together. The developer spent at least 2 full days using EC2, VPC, CodeDeploy, IAM settings under the platform which is necessary to understand before setting up the environment.

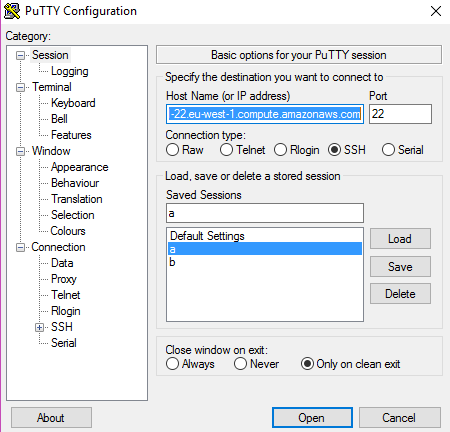
<https://eu-west-1.console.aws.amazon.com/vpc/home>

<https://eu-west-1.console.aws.amazon.com/ec2>

Let us assume that the instance is up and running, there is also another ceremony for you to get connected to the Ubuntu server using a terminal.

You must generate a ppk from your pem file given to you by AWS using PuttyGen.

Load your ppk key to Putty and save it as a profile, with the user format something like; [ubuntu@ec2-52-212-251-22.eu-west-1.compute.amazonaws.com](mailto:ubuntu@ec2-52-212-251-22.eu-west-1.compute.amazonaws.com)



Once you are connected to the remote Ubuntu server you must get the environment ready to install the Django app with the following commands through terminal:

Sudo apt-get update

Sudo apt-get python-pip

Sudo apt-get git-core

Sudo pip install --upgrade pip

git clone https://github.com/NurErtem/Fall2016Swe573NurErtem.git

cd Fall2016Swe573NurErtem/server1/engine

now you need to prep your virtual environment:

pip install virtualenv

virtualenv --python=python3 venv

source venv/bin/active

pip install Django

pip install djangorestframework

pip install requests

python manage.py migrate

also make sure that settings.py under root includes the IP of the AWS instance.

sudo venv/bin/python manage.py runserver 0.0.0.0:88

#the port you may have specified under AWS EC2 console

apt-get install supervisor

create a file name engine.conf with the below text and save it under /etc/supervisor/conf.d/

[program:karmacircle]

command=/home/ubuntu/ Fall2016Swe573NurErtem/engine/venv/bin/python /home/ubuntu/ Fall2016Swe573NurErtem/engine/manage.py runserver 0.0.0.0:88

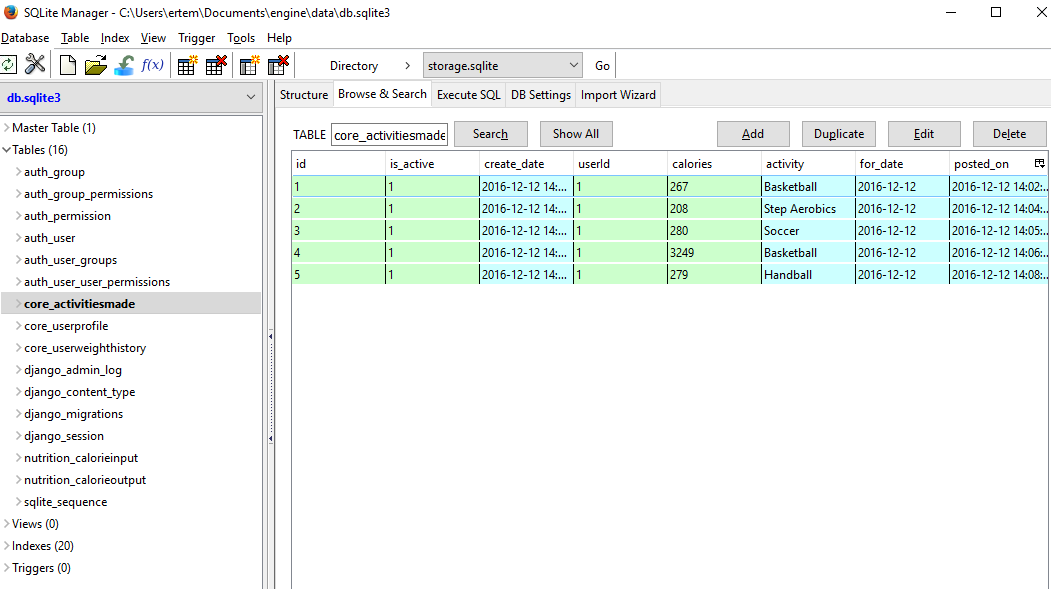
stdout\_logfile=/var/log/supervisor/engine.log

stderr\_logfile=/var/log/supervisor/engine.error

# Test Documents

### Simple UX and data storage test

The web app is able to store activities in the sqlite3 database. It can be tested by running the app locally and checking the sqlite3 manager and reading the entries from the local copy as follows:



# Tools and technologies are learned from scratch and utilized:

* Python 3: the language of the Django framework
* Django 1.10.4: web framework
* SQLite: default DB of Django is used
* Zen Hub: time management tool
* MS Project 2016: Project management tool
* GitHUb tags, issues utilized as tasks, milestones in line with the project plan
* DrawIO: mockups and SDD diagrams
* PyCharm: IDE for Django environment back end development
* WebStorm: IDE for HTML, Javascript,CSS, ajax etc front end development
* Postman: RESTful API browser
* GitHub Desktop: to push and pull local dev files to github
* PIP 9.0.1: pyhton package index
* GIT: command line interface for version control
* PuttyGen SSH key generator utility: AWS pem to pkk conversion on windows
* Putty: connecting to the AWS web server
* WinSCP: synch files between AWS and local
* AWS web interface: deployment environment: EC2, VPC, IAM, CODEDEPLOY
* AWS Command Line Interface
* Under AWS: Ubuntu 16.04
* Under AWS: Apache2
* Virtualenv for Django