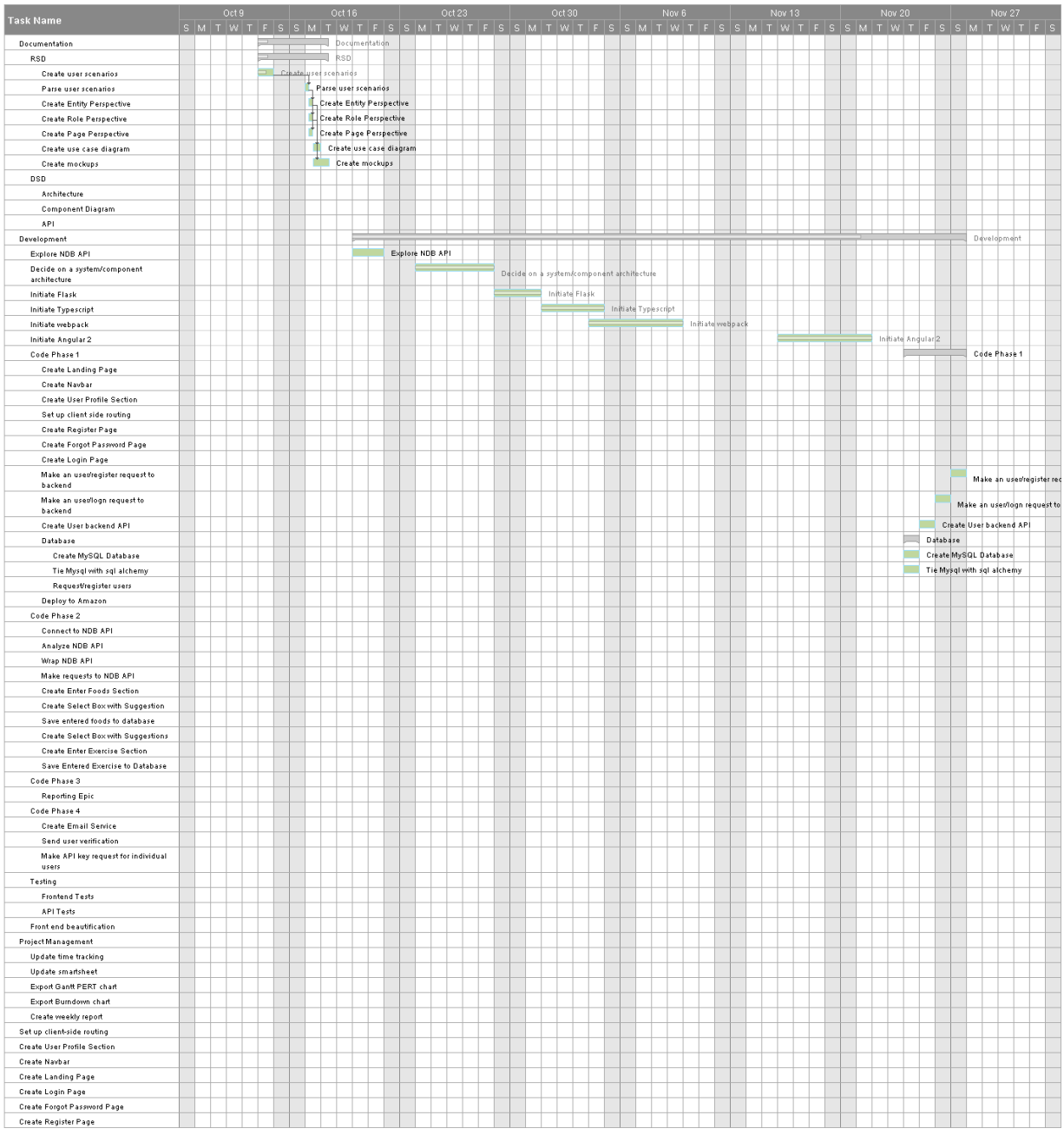
November Worklog

Client	Project	Registered time	Billable time	Amount ()
	Swe 573	31:20:37		



Task Name	Start Date	End Date	Duration	Predecessors	% Comple
Documentation	**10-14-16**	**10-18-16**	**2.5d**		

RSD	10-14-16	10-18-16	2.5d		
Create user scenarios	10-14-16	10-14-16	1d		
Parse user scenarios	10-17-16	10-17-16	0.25d	3	
Create Entity Perspective	10-17-16	10-17-16	0.25d	4	
Create Role Perspective	10-17-16	10-17-16	0.25d	4	
Create Page Perspective	10-17-16	10-17-16	0.25d	4	
Create use case diagram	10-17-16	10-17-16	0.5d	5, 6	
Create mockups	10-17-16	10-18-16	1d	6	
DSD					
Architecture					
Component Diagram					
API					
Development	10-20-16	11-27-16	27d		
Explore NDB API	10-20-16	10-21-16	2d		
Decide on a system/component architecture	10-24-16	10-28-16	5d		1
Initiate Flask	10-29-16	10-31-16	2d		1
Initiate Typescript	11-01-16	11-04-16	4d		1
Initiate webpack	11-04-16	11-09-16	4d		1
Initiate Angular 2	11-16-16	11-21-16	4d		1
Code Phase 1	11-24-16	11-27-16	2d		
Create Landing Page					
Create Navbar					
Create User Profile Section					
Set up client side routing					
Create Register Page					
Create Forgot Password Page					
Create Login Page					
Make an user/register request to backend	11-27-16	11-27-16	1d		
Make an user/logn request to backend	11-26-16	11-26-16	1d		
Create User backend API	11-25-16	11-25-16	1d		
Database	11-24-16	11-24-16	1d		
Create MySQL Database	11-24-16	11-24-16	1d		
Tie Mysql with sql alchemy	11-24-16	11-24-16	1d		
Request/register users					
Deploy to Amazon					
Code Phase 2					
Connect to NDB API					
Analyze NDB API					
Wrap NDB API					
Make requests to NDB API					
Create Enter Foods Section					
Create Select Box with Suggestion					

- **Save entered foods to database**
- **Create Select Box with Suggestions**
- **Create Enter Exercise Section**
- **Save Entered Exercise to Database**

Code Phase 3

- **Reporting Epic**

Code Phase 4

- **Create Email Service**
- **Send user verification**
- **Make API key request for individual users**

Testing

- **Frontend Tests**
- **API Tests**

Front end beautification

Project Management

Update time tracking

Update smartsheet

Export Gantt PERT chart

Export Burndown chart

Create weekly report

Set up client-side routing	2016-11-27T20:45:52Z	2016-12-05T00:00:00Z
Create User Profile Section	2016-11-27T20:44:44Z	2016-12-05T00:00:00Z
Create Navbar	2016-11-27T20:43:54Z	2016-12-05T00:00:00Z
Create Landing Page	2016-11-27T20:41:59Z	2016-12-05T00:00:00Z
Create Login Page	2016-11-27T20:47:44Z	2016-12-05T00:00:00Z
Create Forgot Password Page	2016-11-27T20:47:21Z	2016-12-05T00:00:00Z
Create Register Page	2016-11-27T20:46:45Z	2016-12-05T00:00:00Z

December Worklog

Client	Project	Registered time	Billable time	Amount ()
	Swe 573	20:41:13		

Abstract

Health Tracker Project is a software development assignment which aims to simulate major parts of development including project management and testing. As part of Bogazici University SWE 573 course we are asked by our instructor, Suzan Uskudarli, who acts as our customer, patron and senior developer, to document, design and develop an application.

This document will analyze the Project Description: Health Tracker[i], explore the application domain, find users of the system, create scenarios, apply object and function parsing on the found scenarios, put out functional requirements with the help of use case diagrams and explanations, explore existing interfaces to be used with the software and create the initial expected user experience by defining mockups.

This document will be used to facilitate reviews with the customer.

This document will define the scope of the work.

Final version of this document will provide a reference to the Design Document.

Introduction

Health Tracker is an everyday use tool which shall support quantifying body wellness. It will automate conversion of consumed foods and sports activities to calories. It will generate several reports and lists based on past data. It aims to help keep track of the past progress so that the user can see the trend of their activities and food consumption.

There will be two phases of development. Items marked with the (Phase 2) tag will be implemented in the second phase.

Overall Description

Main metrics to be used in the software are weight, height, food consumption, activity type and activity duration.

Software shall automate the conversion of food consumption to calories, activity type and duration to expanded calories, weight and height to body mass index[1].

Software shall produce reports based on given time intervals. Smallest time unit for dates is day.

Constraints

Nutrition suggestions we receive from NDB API states that these values are not suitable for use of children under the age of 4 and pregnant women.

NDB API limits the requests to 10000 for each token. It is assumed that individual users shall retrieve individual API tokens with their email. Spam controls shall be placed for API requests.

External Software Interfaces

NDB API

https://ndb.nal.usda.gov/ndb/doc/index#

[1] https://en.wikipedia.org/wiki/Body_mass_index

http://www.fda.gov/Food/GuidanceRegulation/GuidanceDocumentsRegulatoryInformation/LabelingNutrition/ucm064928.htm

N4ZFGssGZVgxc8ZtBOp11B1pRPkXj57IsvKcGpvL

Account Email: can.tuksavul@gmail.com
Account ID: 1b66575b-b417-46e7-ab4a-6317852bae1c

https://api.data.gov/nrel/alt-fuel-stations/v1/nearest.json?**api_key=N4ZFGssGZVgxc8ZtBOp11B1pRPkXj57IsvKcGpvL**&location=Denver+CO

User Scenarios

Scenario Brief Descriptions

1. **Scenario 1: Registration, Automatic Login, First Food Entry, Automatic report update**
   1.1. User registers to the system. User is automatically logged into the system.
   1.2. On first time entry user fills the profile form by entering weight, height, date of birth, gender and notes.
   1.3. Upon entering required fields, user is directed to main view.
   1.4. User enters MCD. A list of suggested foods will appear and user chooses MCDONALD'S Hamburger for today.
   1.5. Report is updated to show the changes.
2. **Scenario 2: Exercise entry and edit**
   2.1. User logs into the system. He/she enters an exercise for the first time with clicking the action field.
   2.2. A dropdown suggestion of all available exercises in the system is prompted for auto-fill. The list is updated as the user types.
   2.3. User then enters the duration.
   2.4. Report is updated with new data.
   2.5. He/she realizes that he/she made a mistake and edits the previously entered exercise.
   2.6. Report is updated to show the new data.
3. **Scenario 3: Searching/Browsing nutrition values (Phase 2)**
   3.1. User logs into the system. He/she would like to browse
4. **Scenario 4: Recipe creation (Phase 2)**
   4.1. User logs into the system.
   4.2. He/she tries to add a food she cooked.
   4.3. She enters the ingredients one by one with specific amounts.
   4.4. He/she highlights the items on the list by clicking 'ctrl' on the keyboard or by clicking one of them and clicking another with shift.
   4.5. Once two or more items is selected, the grayed out 'Create recipe' button next to the Food List becomes clickable.
   4.6. He/she enters a name of the recipe. Clicks 'Create recipe'.
   4.7. He/she should be prompted to change the name if a same name recipe or food exists, otherwise the list is updated to contain a new item. This item should be expandable to show the contents of the recipe.
   4.8. A recipe can't contain other recipes so that we will not deal with recursion. This enforces additional logic during food suggestion to an existing recipe. So in the first version of the software shall not contain editing a recipe.
5. **Scenario 5: Editing a recipe (Phase 2)**
6. **Scenario 6 Viewing advanced Reports (Phase 2)**

6.1. User logs into the system. He/she likes to view advanced reports.

7. Scenario 7: Setting a goal (Phase 2)

7.1. User logs into the system. He/she likes to view past excersizes.
7.2. User logs into the system. He/she likes to view past foods.
7.3. User logs into the system. He/she likes to find a recipe he/she previously entered.
7.4. User logs into the system. He/she sets a goal calorie.
7.5. User logs into the system he/she sets a weight goal.

Scenario Detailed Descriptions

Scenario 1 Details:

1. Scenario 1: Registration, Automatic Login, First Food Entry, Automatic report update

1.1. User registers to the system. User is automatically logged into the system.
1.1.1. Credentials shall be remembered. Cookies shall be used. It might be a good idea to prompt the user that we are using cookies and it might even be better to show a link of our cookie policy where we promise we shall never allow an ad network token shall be placed in the cookies but we will be adding google analytics.
1.1.2. Authentication shall be timed out unless it is inside a mobile webview.
1.2. On first time entry user fills the profile form by entering weight, height, date of birth, gender and notes.
1.2.1. User shall be notified/hinted that by entering these details a report of their current state will be generated.
1.2.2. A user is said to be entering for the first time if one of the mandatory user profile data is missing.
1.2.3. Fields other than Notes shall be mandatory. (only optional field)
1.2.4. User should not be able to continue without entering mandatory fields.
1.3. Upon entering required fields, user shall be directed to main view.
1.3.1. Main view shall consist of current status report, weight, food and activity entry forms.
1.3.1.1. Reports
1.3.1.1.1. First report shall show BMI and expected calorie intake and expenditure of a user based on user profile
1.3.1.2. Second report shall show weekly calorie intake/expenditure wave graph.
1.3.1.3. Link to detailed reports.
1.3.1.4. Daily fulfilled nutrient values. Entered food values should be subtracted from advised adult intake. We will have a report
1.3.1.5. Food entry form
1.3.1.5.1. Food name field should be a writable selection box with auto complete. Required.
1.3.1.5.2. Food amount text field. Required.
1.3.1.5.3. Food amount unit: [kg, gr, ml, lt, quantity]. Required. Imperial is not supported. Defaults to grams.
1.3.1.5.4. Date: defaults to Today.
1.3.1.6. Activity entry form
1.3.1.6.1. Activity Name: Given that activities shall be listed; it is a writable selection box with auto complete. Required.
1.3.1.6.2. Activity Duration: Field that takes a time interval. If user enters a value to this field, Direct Calorie expenditure field shall be grayed out.
1.3.1.6.3. Activity duration unit: [hours, minutes]. Required. Defaults to minutes.
1.3.1.6.4. Direct Calorie expenditure: If user enters a value to this field, Activity Duration field shall be grayed out.

1.3.1.6.5. Date: defaults to Today.

1.3.1.7. Weight entry form

1.3.1.7.1. Weight. Required.

1.3.1.7.2. Weight Unit: [kg/gr]. Required.

1.3.1.7.3. Date: defaults to Today

1.4. User enters MCD. A list of suggested foods will appear and user chooses MCDONALD'S Hamburger for today. Report is updated to show the changes.

1.4.1. Upon entering 3 letters a dropdown list shall be generated.

1.4.1.1. Dropdown list will have 3 sub-sections.

1.4.1.1.1. First section shall list the frequently used items.

1.4.1.1.2. Second section shall list the items created by the user. (Phase 2)

1.4.1.1.3. Third section shall list the remaining food entries from the NDB. It is expected to receive a long list from NDB, so the drop down-list shall not close on losing focus and dropdown list shall be scrollable. We have limited API requests available so every user shall have a specific API token retrieved from the NDB API.

1.5. Report is updated to show the changes.

1.5.1. All data that the reports depend on shall be updated on new entries. This is the main challenge of Phase 1.

2. Scenario 2: First exercise entry

2.1. User logs into the system. He/she enters an exercise for the first time with clicking the action field.

2.2. A dropdown suggestion of all available exercises in the system is prompted for auto-fill. The list is updated as the user types.

2.3. User selects hiking.

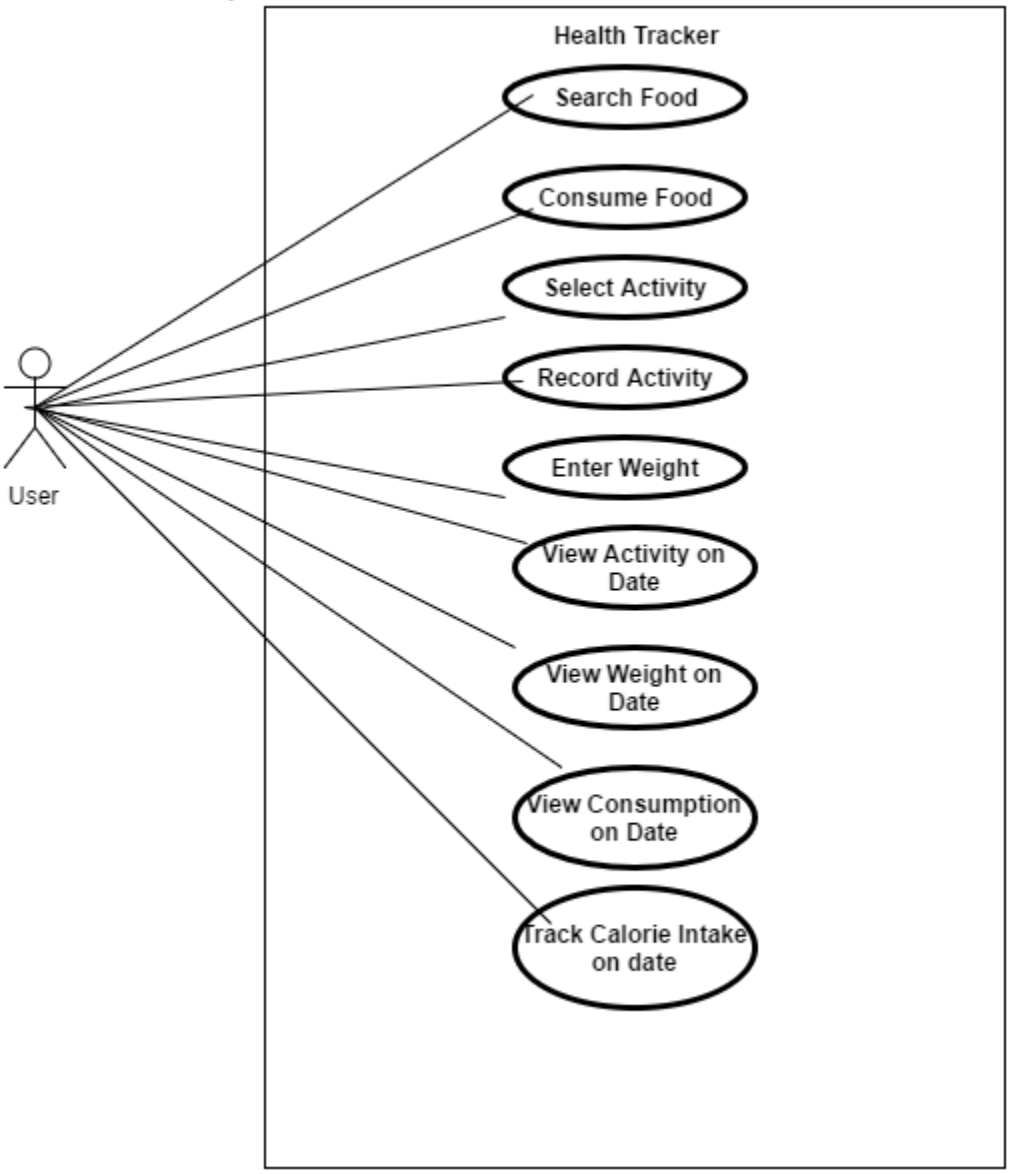
2.4. User enters the duration.

2.5. Report is updated with new data.

2.6. He/she realizes that he/she made a mistake by entering hiking because he/she didn't know that walking was available. He/she edits the previously entered hiking exercise.

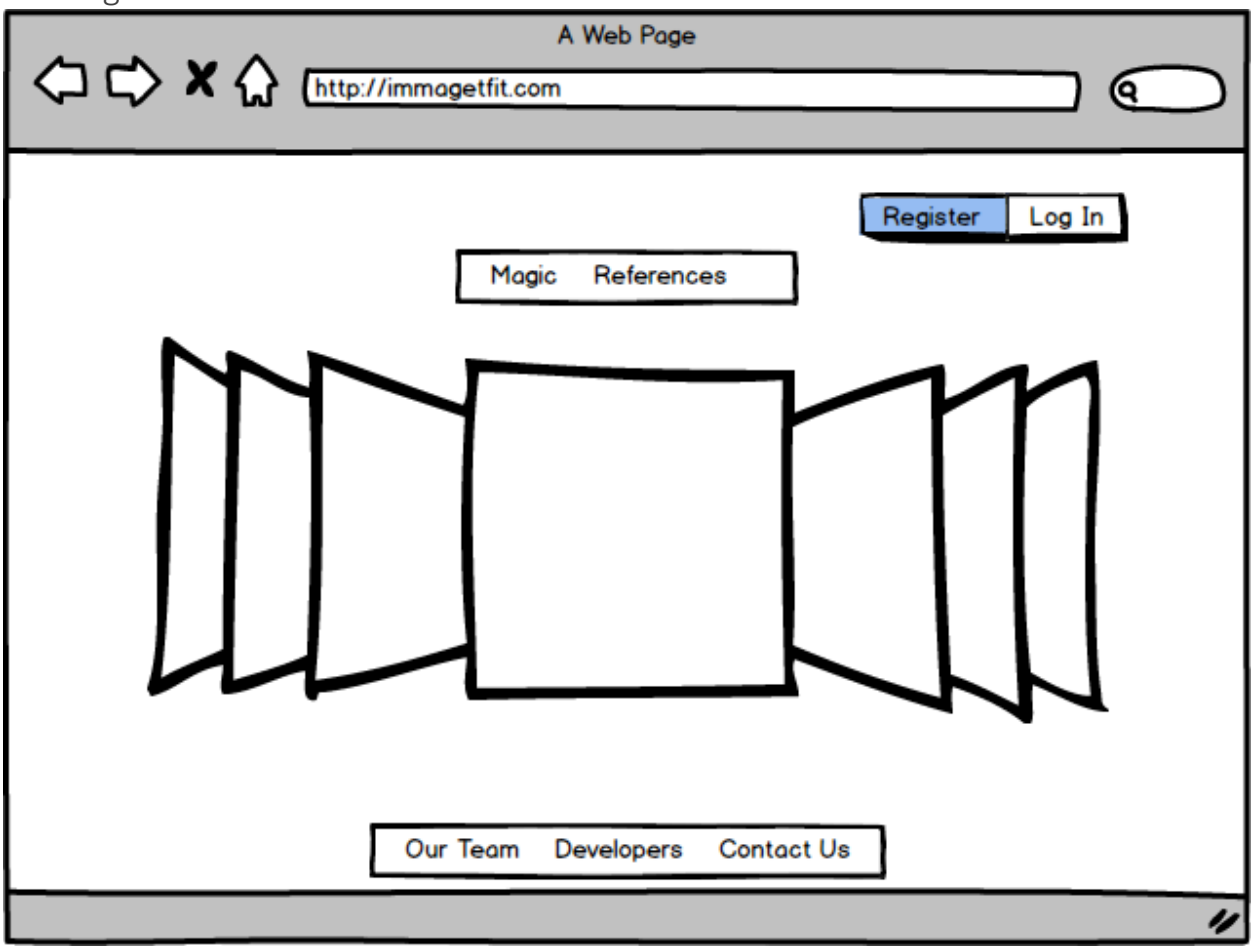
2.7. Report is updated to show the new data.

Use Case Diagram



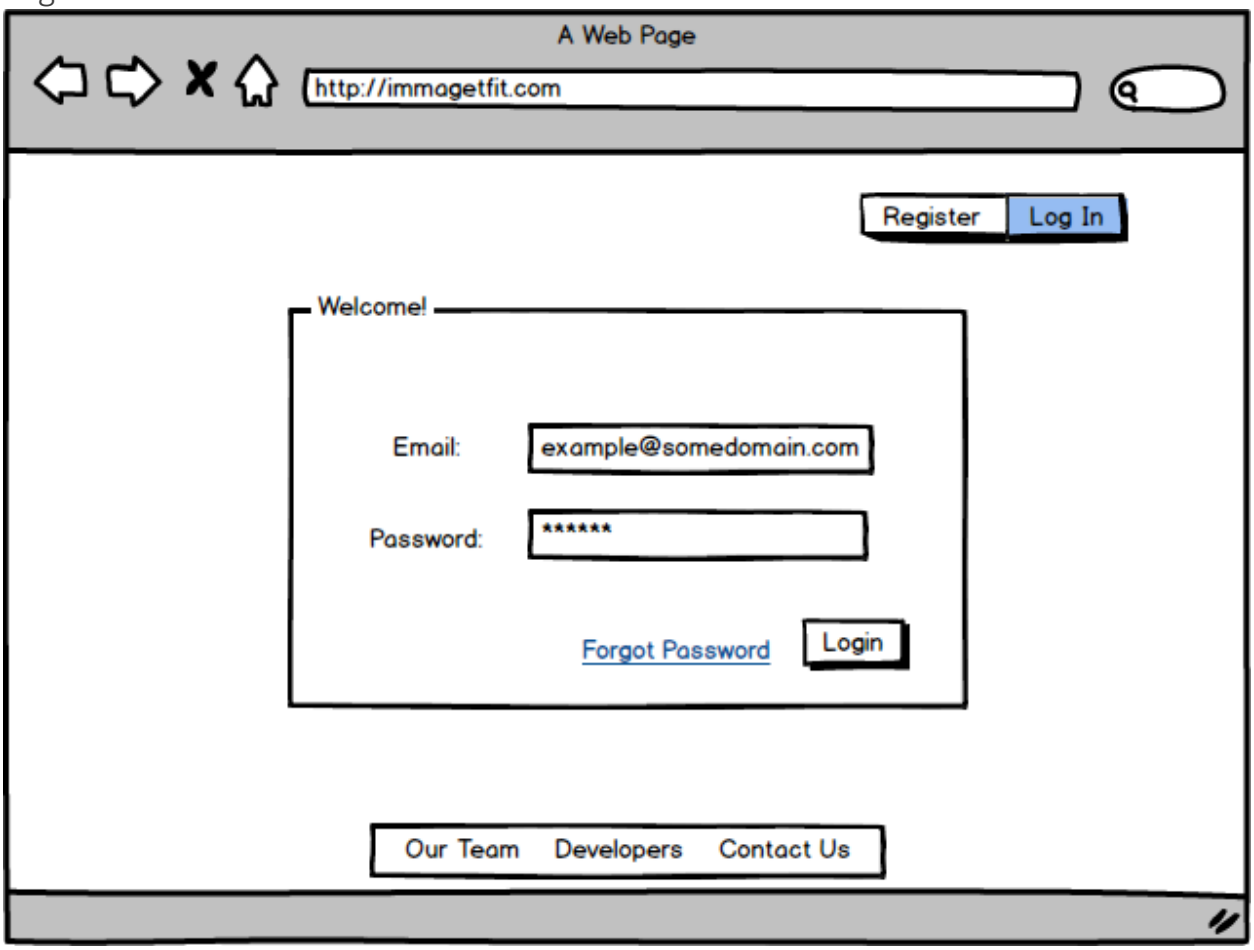
Views List

Landing



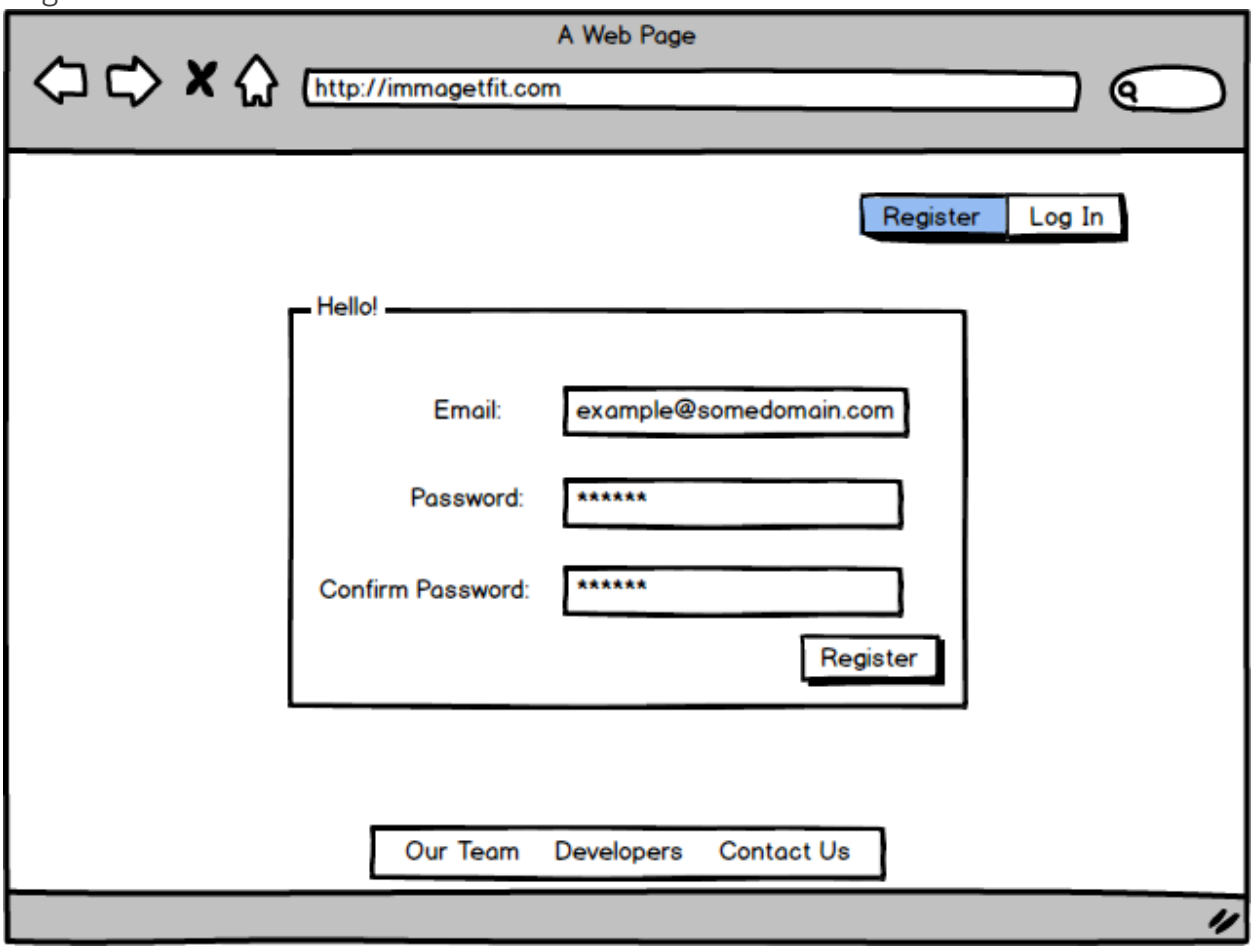
Landing page should contain a brief description of the software. Show some uses of the application and maybe a video of how to use the application and how it can help its users.

Login



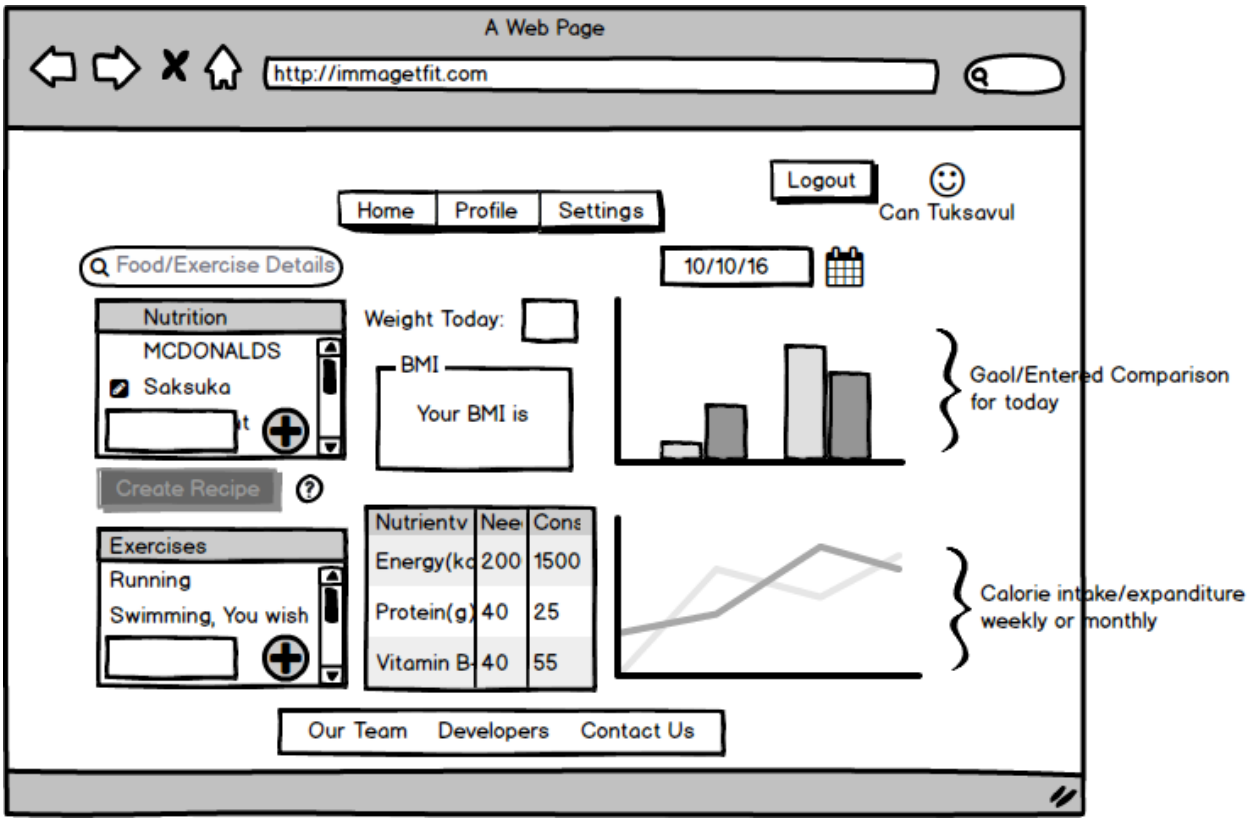
Only email will be used for identification.

Register



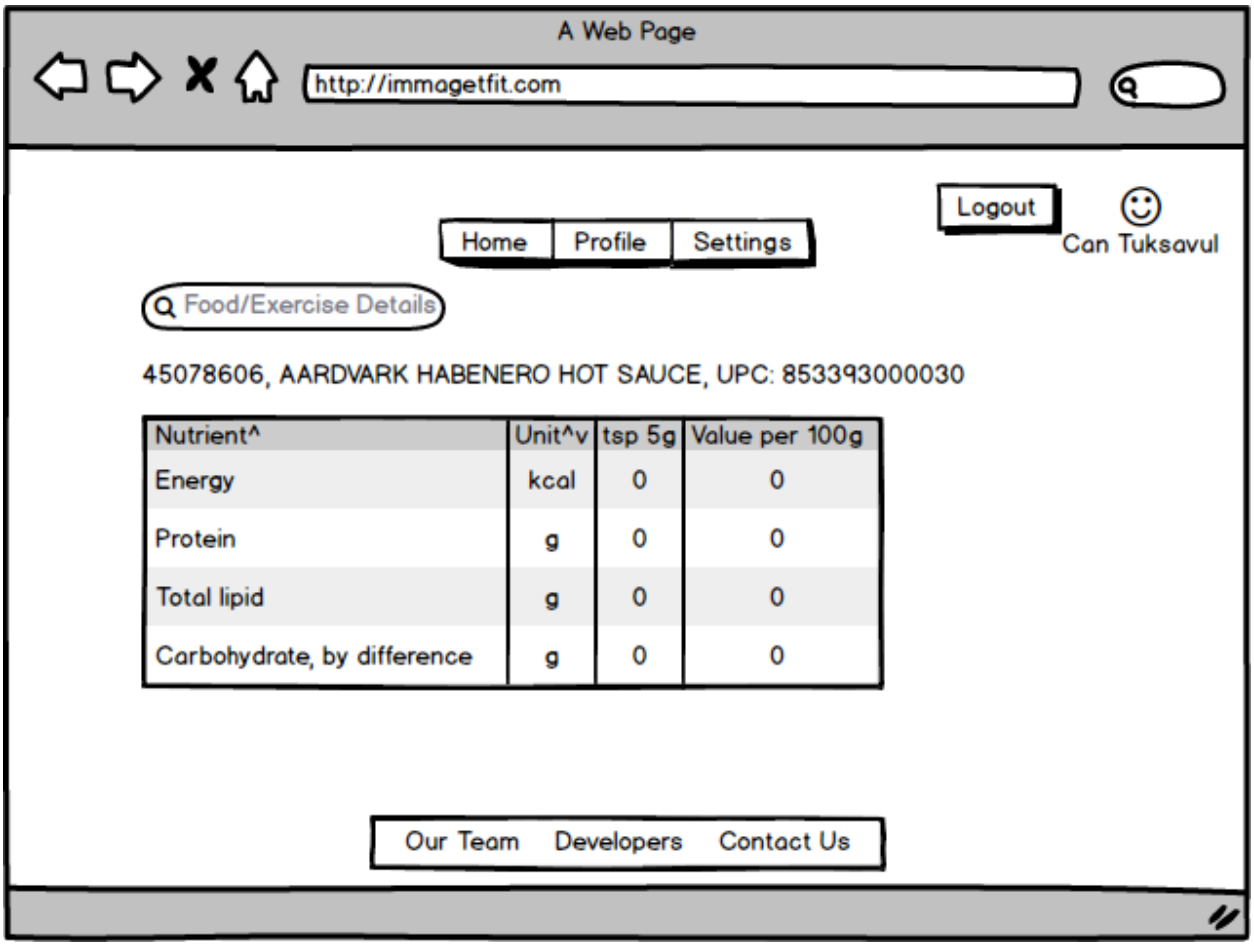
Registering should be easy and fast. Register should redirect user to the home page immediately as we would like to show our goods to lure the user.

User Home



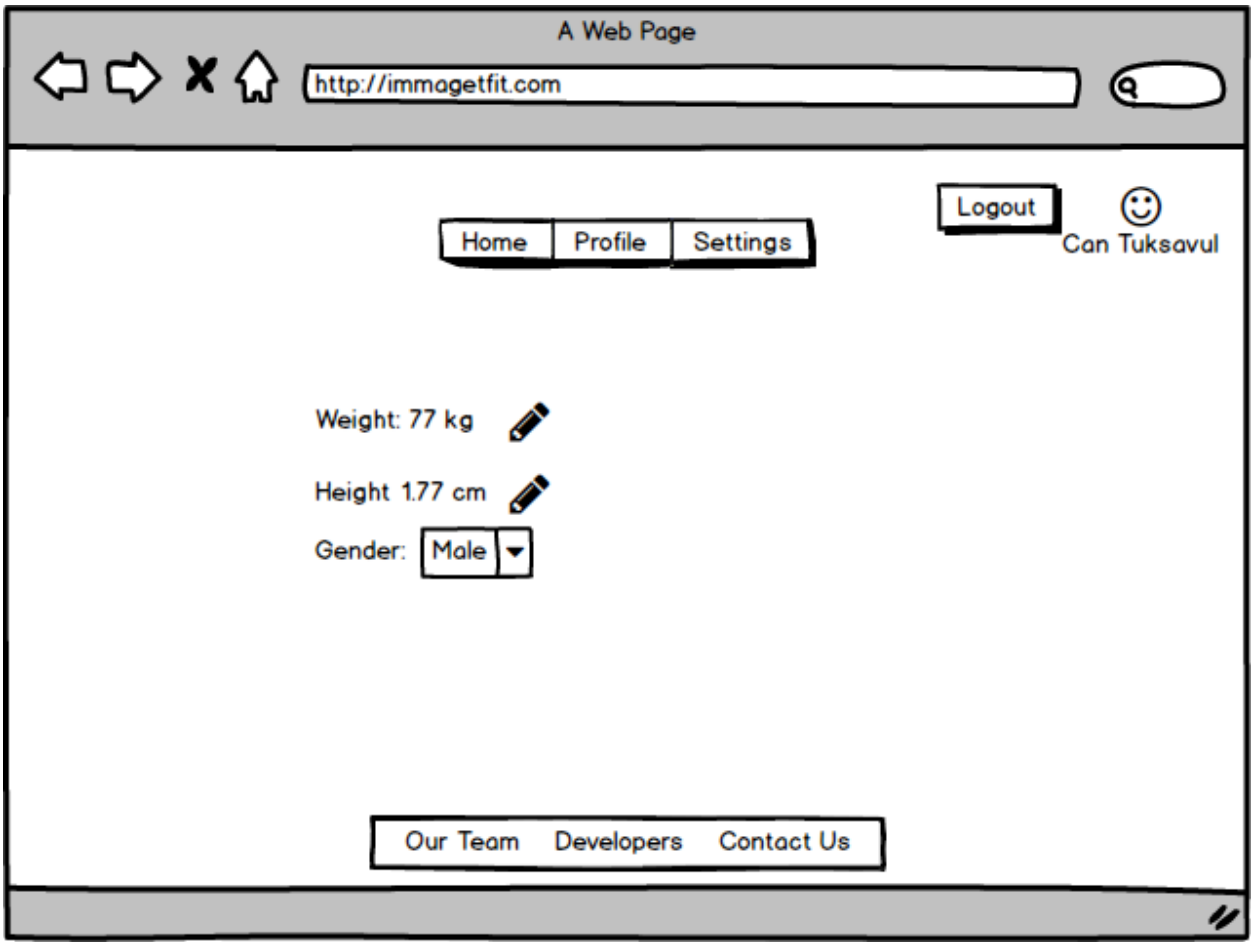
This is the main application interface of the software. User is expected to spend most of his/her time in this page. This page contains the most important functionalities of the software such as entering activity, food or weight and tracking it on a daily basis.

Search Food

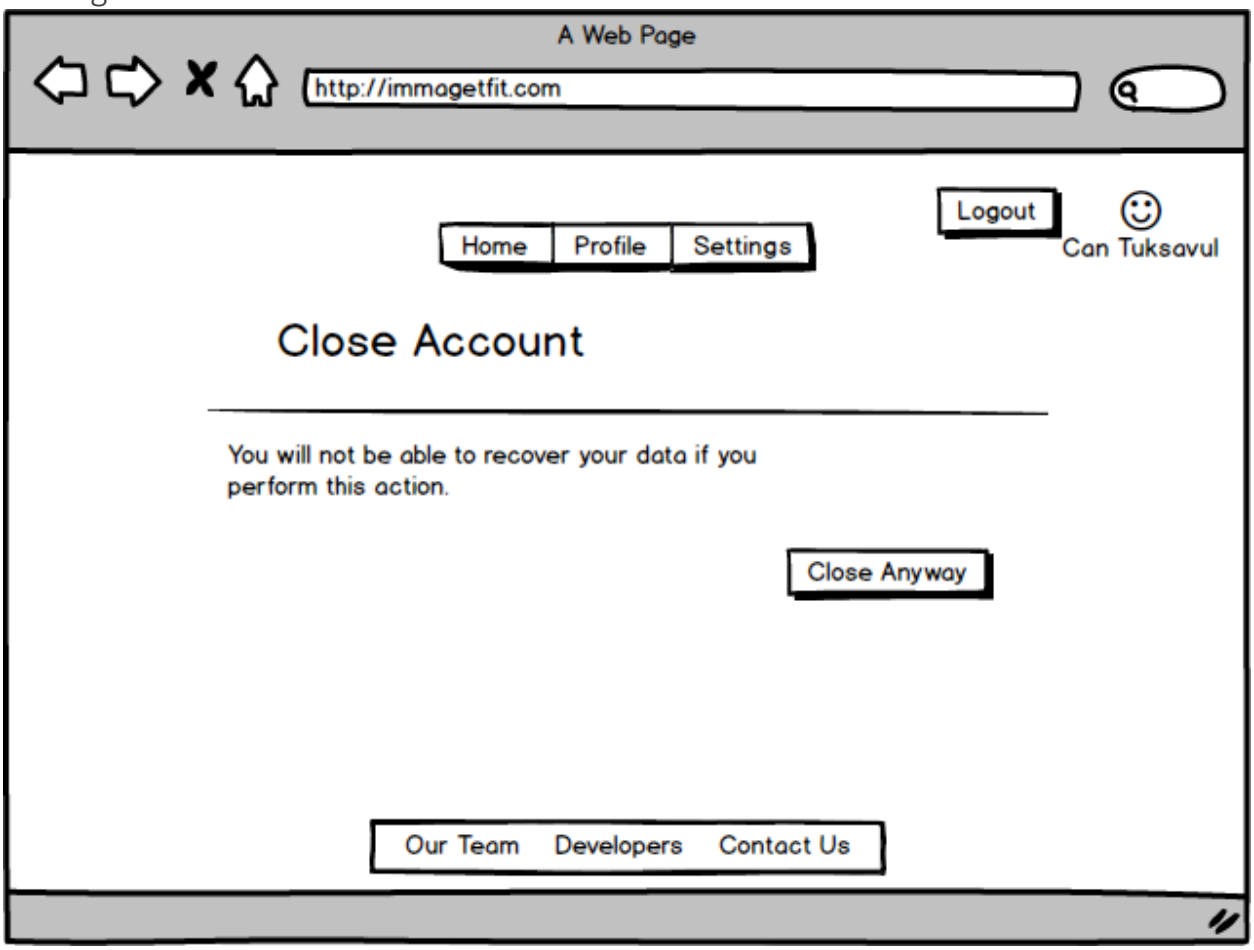


NDB API provides a big dataset. Being able to easily search for nutritional values of the food adds a decent value to the application. One can easily imagine users lengthily fiddling with the application by searching different foods.

Profile


A Web Page
http://immagetfit.com
Logout
Can Tuksavul
Home
Profile
Settings
Weight: 77 kg
Height 1.77 cm
Gender:
Male
Our Team
Developers
Contact Us

Settings



Questions

Where to get the data of which activity will burn how much calorie?

We will be digging for this

Where to get the data of a normal calorie intake for given user profile?

A reference to this is in the project description document

How should the system handle missing data? (this requires that the reports to be generated with sparse data) example: User enters data on Monday, does not enter any data for Tuesday, enters data for Wednesday. In such a case system does not know the total intake for a given time interval.

Reports will be discrete

Contents

Introduction 17
Architecture 17
Client Side Architecture 18
Components 18
Client Components 18
API 19
Test 19
Login 19
Register 19
Update User Details 19
Get the User Details 19
Get Activities 19
New Activity 19
Search NDB API 19
Get NDB Food 19
Create Consumption 19
Get All Foods in a Given Date 19
Get Calorie Intake Between Given Dates 20
Get Total Nutrition Values on a given Date 20
Create/Update Weight on a Given Date 20
Get Weight on a Given Date 20
Get Weights between a given date interval 20
References 20

Introduction

This document triesr to capture design elements of the Health Tracker software produced for the Software Development Practice course in Bogazici University Software Engineering Department.

Main focus will be on transferring design idea through visual representations and their supporting textual descriptions.

Highest level representations will be the following: Deployment Diagram, Component Diagram

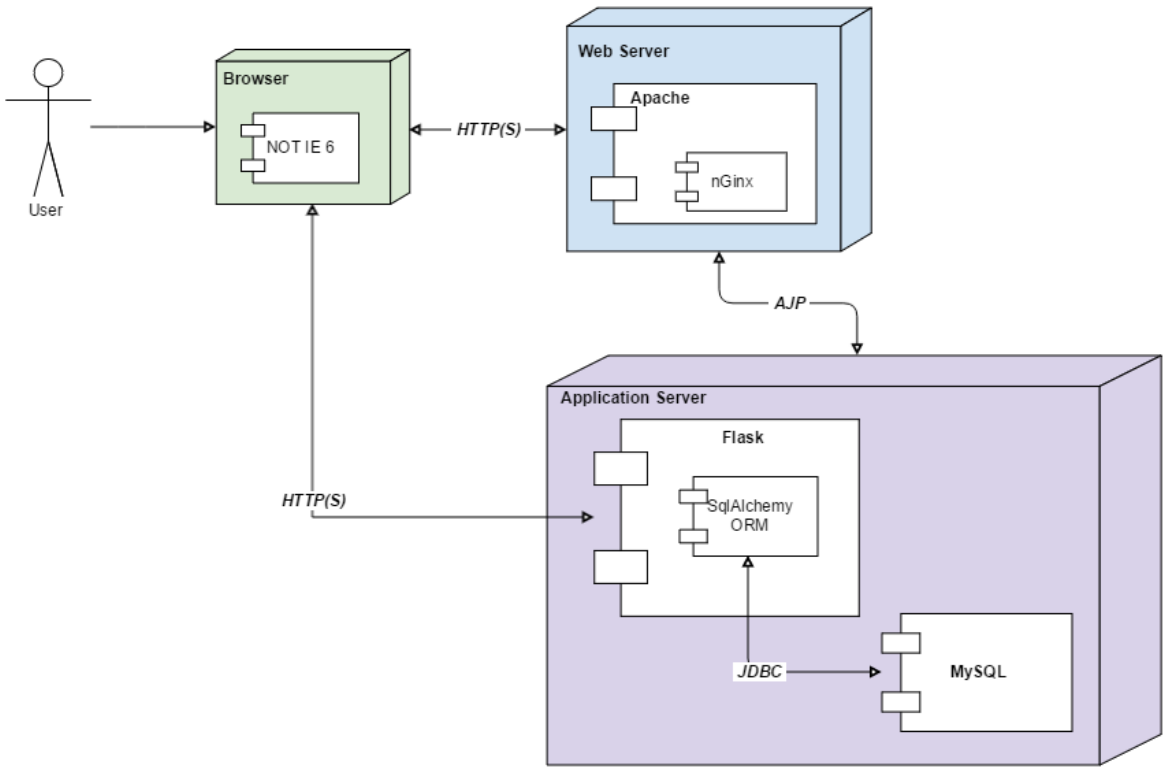
ER Diagram is used for showing the application's main assets.

Rest of the representations focus on providing implementation design view by using the following: Sample Class Diagram, Sample Sequence Diagram.

Server Backend API will be provided as it is sort of the backbone of the application and it also shows parallelism to use cases as well as it can be used for providing entry points to both backend and frontend applications.

Architecture

Server Client architecture is used. Client uses MVC architecture. Server is designed to be a RESTful service. Datastore communications are handled through Object Relational Mapping.



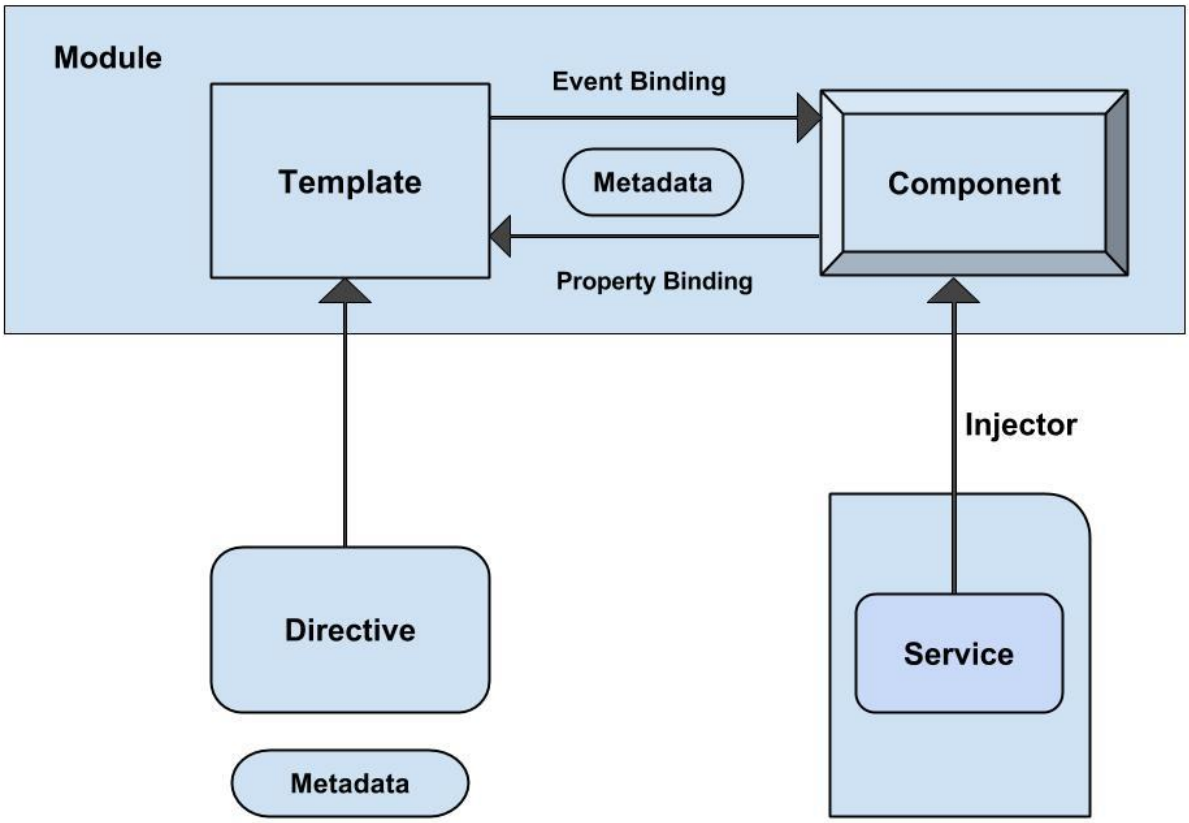
Client Side Architecture

Following is one of the best descriptions of the Angular 2 architecture. Reference: (tutorialspoint)

Angular applications consists of modules and components. Modules bundles components, pipes, services and handles dependency injection.

Components are generally act as views and controllers together of the traditional Model View Controller pattern.

Services act similar to Models but I also chose to add abstract classes of the backend models in a separate folder to increase readability.



Components

Components is shown in two sections.

Client Components

API

Test

Simple test url. Calling this should return an empty template with “Hello World” text.

url: ht.cantuksavul.com:5000

Login

Logs in with the UserCredential model credentials.

url: ht.cantuksavul.com:5000/api/auth', methods=['POST']

Register

Registers a new user with the UserCredential model credentials.

url: ht.cantuksavul.com:5000/api/auth/new', methods=['POST']

Update User Details

Updates user details with the UserDetails model.

url: ht.cantuksavul.com:5000/api/user/<int:user_id>', methods=['PUT']

Get the User Details

Returns UserDetail with the given Id.

url: ht.cantuksavul.com:5000/api/user/<int:user_id>', methods=['GET']

Get Activities

Returns Activity[] model for the given date

url: ht.cantuksavul.com:5000/api/user/<int:user_id>/activity/<date>', methods=['GET']

New Activity

Creates new activity for the given date.

url: ht.cantuksavul.com:5000/api/user/<int:user_id>/activity/new', methods=['POST']

Search NDB API

Searches chars from NDB API.

url: ht.cantuksavul.com:5000/api/food/<chars>', methods=['GET']

Get NDB Food

Returns the Food model of the food with the given ndbno.

url: ht.cantuksavul.com:5000/api/food/<ndbno>/measures', methods=['GET']

Create Consumption

Creates a consumption with the Food model.

url: ht.cantuksavul.com:5000/api/user/<int:user_id>/food/new', methods=['POST']

Get All Foods in a Given Date

url: ht.cantuksavul.com:5000/api/user/<int:user_id>/food/<date>', methods=['GET']

Get Calorie Intake Between Given Dates

url: ht.cantuksavul.com:5000/api/user/<int:user_id>/consumption/energy/<start_date>/<end_date>', methods=['GET']

Get Total Nutrition Values on a given Date

url: ht.cantuksavul.com:5000/api/user/<int:user_id>/consumption/<date>', methods=['GET']

Create/Update Weight on a Given Date

Using Weight model

url: ht.cantuksavul.com:5000/api/user/<int:user_id>/weight/<selected_date>', methods=['PUT']

Get Weight on a Given Date

Using Weight model

url: ht.cantuksavul.com:5000/api/user/<int:user_id>/weight/<selected_date>', methods=['GET']

Get Weights between a given date interval

Using weight[] model.

url: ht.cantuksavul.com:5000/api/user/<int:user_id>/weight/<start_date>/<end_date>', methods=['GET']

References

tutorialspoint. (n.d.). Retrieved from https://www.tutorialspoint.com/angular2/angular2_architecture.htm

Contents

Development Platform 22
Development Tools 22
Deployment Platform 22
Software Libraries & Frameworks Used 22

Development Platform

I used Debian Jessie x86 as the developments platform.

Development Tools

Javascript IDE: Webstorm

Python IDE: Pycharm

MySql Editor: Mysql workbench

Build & Testing: Webpack-dev-server

Browser: Chrome Browser

Python package manager: Anaconda

Node package manager: NPM

Deployment Platform

Amazon EC2 Ubuntu 14.04 ABI

Mysql-server

Httpd

Miniconda

NPM

Software Libraries & Frameworks Used

Python Flask

Flask-CORS

Flask-SQLAlchemy

Angular 2 Typescript

Webpack module bundling and builds

Protractor testing

Karma testing

Jasmine Testing

PrimeNG for UI components.

Deployment Commands

```
#!/bin/sh


apt-get update

apt-get upgrade

apt-get install git
```

wget https://repo.continuum.io/miniconda/Miniconda2-latest-Linux-x86_64.sh

bash Miniconda2-latest-Linux-x86_64.sh

apt-get install mysql-server

sudo apt-get install libmysqlclient-dev

apt-get install npm

sudo npm install -g n

sudo n stable

git clone https://github.com/cantux/Fall2016Swe573cant.git

cd Fall2016Swe573cant/code/public

npm install webpack-dev-server rimraf webpack -g

npm install

npm run build:prod

cp -R dist/* /var/wwww/html/.

mysql -u root -p admin -s "CREATE DATABASE deneme2;quit;"

cd ../app

conda env create swe573_backend -f backend.env.yml

source activate swe573_backend

export FLASK_APP=FlaskDeneme.py

nohup flask run --host=0.0.0.0

Frontend Testing

General Tests Infrastructure: The Jasmine test framework. provides everything needed to write basic tests. It ships with an HTML test runner that executes tests in the browser.

Unit tests: Karma – The karma test runner is ideal for writing and running unit tests while developing the application. It can be an integral part of the project's development and continuous integration processes. This guide describes how to setup and run tests with karma.

Karma has a simple api that defines javascript testing infrastructure. The most basic functions are: it() expect() and describe(). Karma was built by the angular team to be able to tes angularjs.

End-to-end tests : Protractor – End-to-end tests explore the application *as users experience it*. In e2e testing, one process runs the real application and a second process runs protractor tests that simulate user behavior and assert that the application responds in the browser as expected.

Testing Examples

Protractor:

```
import { browser, by, element } from 'protractor';

describe('App', () => {
  beforeEach(() => {
    browser.get('/');
  });
  it('should have header', () => {
    let subject = element(by.css('h1')).isPresent();
    let result  = true;
    expect(subject).toEqual(result);
  });
});
```

Karma

```
import {
  inject,
  TestBed
} from '@angular/core/testing';

// Load the implementations that should be tested
import { AppComponent } from './app.component';
import { AppState } from './app.service';

describe('App', () => {
  // provide our implementations or mocks to the dependency injector
  beforeEach(() => TestBed.configureTestingModule({
    providers: [
      AppState,
      AppComponent
    ]}));

  it('should have a url', inject([ AppComponent ], (app: AppComponent) => {
```

```
    expect(app.url).toEqual('https://twitter.com/AngularClass');
  }));

});
```

[i] https://github.com/cantux/Fall2016Swe573cant/blob/master/docs/SWE573_projectdescription.pdf