Functional Spec

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Description of change | Author | Date |
| 1 | Initial documentation | Felix Okoth | 15th April 29, 2017 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[Introduction 3](#_Toc481313873)

[Purpose 3](#_Toc481313874)

[Scope 3](#_Toc481313875)

[Out of scope 4](#_Toc481313876)

[Background 5](#_Toc481313877)

[References 5](#_Toc481313878)

[Assumptions and constraints 5](#_Toc481313879)

[Assumptions 5](#_Toc481313880)

[Constraints 6](#_Toc481313881)

[Document Overview 6](#_Toc481313882)

[Methodology 7](#_Toc481313883)

[Functional Requirements 8](#_Toc481313884)

[Authenticating users 8](#_Toc481313885)

[Registering users 9](#_Toc481313886)

[Log-In 10](#_Toc481313887)

[Update user profile 11](#_Toc481313888)

[User Requirements 11](#_Toc481313889)

[Data Flow Diagrams 12](#_Toc481313890)

[Logical Data Model 12](#_Toc481313891)

[Functional Requirements 13](#_Toc481313892)

[Functional Requirements Groups 15](#_Toc481313893)

[Other Requirements 21](#_Toc481313894)

[Interface Requirements 21](#_Toc481313895)

[Data Conversion Requirements 21](#_Toc481313896)

[Hardware/ Software Requirements 21](#_Toc481313897)

[Operational Requirements 21](#_Toc481313898)

# Introduction

Introduce here.

## Purpose

The purpose of this document is to lay out the functional specification of …..

## Scope

The areas to be covered will be: -

1. User registration: Registration of new users will collect and store basic user information such as name, email, photo, gender, date of birth, weight, height and preferred location. In addition, the token-based authorization id, will be collected based on the OAuth provider that the user would prefer to use to authenticate into the system.
2. OAuth authentication: The system will support authentication using OAuth services that are provided by Facebook, Google, Twitter, Git and Yahoo.
3. User profile update: It will be possible for the user to update their profile with new information i.e. change of name, profile picture, weight etc.
4. There will be Api’s that returns a list of countries and languages that are supported by the system.

## Out of scope

Password related operations (set/reset etc.)

Localization: Only U.S English language will be supported in the initial stage. The development will be done with localization in mind to make it easier to adopt to other regions/languages when the time comes.

Only video media will be supported [TODO] list supported video types and reasons for supporting those types and leaving out the rest.

All requests will be done through email/sms communication. System built notifications for requests will be considered in the next version.

UI, UX, Deployment, Test and System Specification documents are out of scope for this work.

[TODO] Specify search limits.

Deployment scripts, deployment strategies, upgrades are out of scope for this document.

## Background

[TODO ] Specify who the customer is and the need.

[TODO PM] Specify who will be in charge of the deployment process, upgrades and general server and keys security.

[TODO PM] Specify the release cycles for the project with respect to dev work,

[TODO PM] Specify the bugs/DCR fix cycles and the project methodology that will be used.

## References

<https://www.ubuntu.com/server>

<http://wildfly.org/>

<http://wildfly-swarm.io/>

<http://www.json.org/>

<https://jersey.java.net/>

<https://oauth.net/>

<https://www.microsoft.com/en-us/sdl/>

<http://www.oracle.com/technetwork/java/javaee/overview/index.html>

Refer to PM doc

Refer to UI doc

Refer to wireframe doc

Refer to Threat model doc

Dev project plan.

## Assumptions and constraints

### Assumptions

It is assumed that the application will purely be accessed through small devices (see UI doc).

Firebase will be used for any notifications work, in addition to email and possibly sms alerts.

A deployment server will be provided that is well tested with all the requirements. The deployment servers will be well secured.

The hardware that will be used to access the system will be geo-location enabled and with the capability to upload videos in the right format.

Access to the backend will only be through jax-rs calls.

It will be possible to integrate any third-party utilities such as for sending sms and emails through java code.

### Constraints

In the initial stages, only small set of users will be able to be onboarded.

## Document Overview

The document is made up of flow charts, DFDs and sample codes that demonstrates the intended implementation strategy.

Within the document is the intended resource paths as will be deployed in the server. UI development works can assume that the resource paths are valid, unless otherwise stated.

# Methodology

The methodology that will be used for development cycles will be determined by the P.M. Microsoft SDL 2016 tools will be used to model security threats and requirements. The threat models will be linked to this document based on architectural plan and software implementation plan.

The P.M will set bug bars for security and privacy to be met by the system. The bugs will have prioritization from 1 to 4, with one being the highest. Severity will be from 1 to 4 as well.

# Functional Requirements

**Context**

### Authenticating users



### Registering users

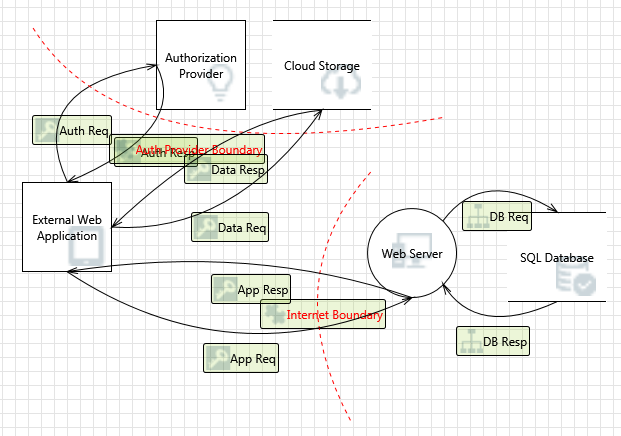
### Log-In

### Update user profile

## User Requirements

1. Server box with at least 4GB RAM. After testing phase is over, the systems will have to run behind a load balancer for both the front ends that would be good enough to handle the anticipated requests.
2. Ssl certificate for with trendingtopit.com
3. The UI will have to make all requests and get all responses through jax-rs.

## Data Flow Diagrams



## Logical Data Model

Below are the data tables that will be implemented in the database to support the system data requirements.

[TODO] Add those data schemas

## Functional Requirements

1. All the return values, other than for authentication are in an object that has: -
   1. Message
   2. Status
   3. Data

**public** **class** TResponse<T> **implements** Serializable {

**private** String message;

**private** **int** status;

**private** T data;

**public** String getMessage() { **return** message; }

**public** **void** setMessage(String message){ **this**.message = message; }

**public** **int** getStatus() { **return** status; }

**public** **void** setStatus(**int** status) { **this**.status = status; }

**public** T getData() { **return** data; }

**public** **void** setData(T data) { **this**.data = data; }

}

1. Authentication returned object that has all the required values and the session key.

**public** **class** TSession **implements** Serializable {

**private** String userName;

**private** String userId;

**private** String clientIp;

**private** String photo;

**private** String key;

**private** **int** status;

**private** String displayName;

**private** String companyName;

**private** String companyLogo;

**private** String id;

**private** String copyright;

**private** OLicenceType olicense;

**private** **boolean** fedauth;

**public** **boolean** isFedauth() { **return** fedauth; }

**public** **void** setFedauth(**boolean** fedauth) { **this**.fedauth = fedauth; }

**public** OLicenceType getOlicense() { **return** olicense; }

**public** **void** setOlicense(OLicenceType olicense) {

**this**.olicense = olicense;

}

**public** String getCopyright() { **return** copyright; }

**public** **void** setCopyright(String copyright) {

**this**.copyright = copyright;

}

**public** String getId() { **return** id; }

**public** **void** setId(String id) { **this**.id = id; }

**public** String getDisplayName() {

**return** displayName;

}

**public** **void** setDisplayName(String displayName) {

**this**.displayName = displayName;

}

**public** **int** getStatus() { **return** status; }

**public** **void** setStatus(**int** status) { **this**.status = status; }

**public** String getKey() { **return** key; }

**public** **void** setKey(String key) { **this**.key = key; }

**public** String getPhoto() { **return** photo; }

**public** **void** setPhoto(String photo) { **this**.photo = photo; }

**public** String getUserName() { **return** userName; }

**public** **void** setUserName(String userName) { **this**.userName = userName; }

**public** String getUserId() { **return** userId; }

**public** **void** setUserId(String userId) { **this**.userId = userId; }

**public** String getClientIp() { **return** clientIp; }

**public** **void** setClientIp(String clientIp) { **this**.clientIp = clientIp;}

**public** String getCompanyName() { **return** companyName; }

**public** **void** setCompanyName(String companyName) {

**this**.companyName = companyName;

}

**public** String getCompanyLogo() { **return** companyLogo; }

**public** **void** setCompanyLogo(String companyLogo) {

**this**.companyLogo = companyLogo;

}

}

1. Manage users

[TODO] Add user object schema

1. List of countries

**public** **class** Country **implements** Serializable {

**private** String code;

**private** String description;

}

1. List of supported languages

**public** **class** Language **implements** Serializable {

**private** **int** lciddec;

**private** String name;

**private** String lcidhex;

**}**

### Functional Requirements Groups

|  |  |
| --- | --- |
| **#** | **Details** |
| FR 2 Authentication | Sign in user using OAuth |
| Sign out user |
| Switch user preferred sign in provider |
|  |
| FR 3 Manage Users | Register user |
| Update user profile |
| Update user picture |
|  |
| FR 4 Countries | List Country |
| FR 5 Languages | List Language |
| Free Advertising[TODO] PM |  |

**Function Requirements Resource Paths**

The base resource path will be:

1. Members [TODO] add the resource path

@POST

@Path("/umemberphoto/{id}")

@Consumes(MediaType.***MULTIPART\_FORM\_DATA***)

@Produces(MediaType.***TEXT\_PLAIN***)

**public** Response setUserPhoto(

**final** MultipartFormDataInput io,

@PathParam("id") **final** String id,

@HeaderParam("AuthorizationKey") String sessionKey){

1. Base: @Path("/base")

@GET

@Path("/lcountry")

@Produces(MediaType.***APPLICATION\_JSON***)

**public** TResponse<List<Country>> listCountry() {

@GET

@Path("/llanguage")

@Produces(MediaType.***APPLICATION\_JSON***)

**public** TResponse<List<Language>> listLanguage() {

## Other Requirements

### Interface Requirements

All the calls to the apis will be through http(s) protocol on port 443.

### Data Conversion Requirements

All data, other than media data will be through json objects. Protobuf is also an alternative that can be considered, if json will prove to have limitations.

### Hardware/ Software Requirements

1. At least Wildfly10.1.0 Final or WildflySwam
2. At least Ubuntu 16.04 LTS server
3. The latest MySql
4. At least a 2 code 1.7 GHz server.
5. At least 4 GB RAM
6. At least 10 GB HD for a start.

### Operational Requirements

1. The system up-time will be pegged at 99.9% up-time
2. Regular data backup will be done
3. The system will be put behind a load balance to be elastic in nature i.e. be able to expand based on the needs at the time and contract when less resources are needed.
4. System restore, in case of irrecoverable data loss will take 1 hour at most.

#### Security and privacy

The security and privacy of user data within the system will be addressed to the highest point possible.

1. A loss or corruption of data can result to lack of trust with the users and thus impact the monetary value of the application.
2. Unintended information disclosure will erode trust with the users thus leading to monetary loss. At the same time, improper handling of PII can lead to charges being brought about by the federal government.

The system will be developed with security in mind from inception, through development, through testing to deployment, upgrade, production, backup, and restore.

1. Physical access to