Whoop-Txt SDD Draft Version 1.0

Software Requirements Specification Prepared By:

Anthony Balmeo, Jessica Kain, Zorayr Khalapyan, Wade Norris, Zachary Policzer, Parth Shah

Table of Contents

[1. Introduction 2](#_Toc307084128)

[1.1 Purpose 2](#_Toc307084129)

[1.2 Scope 2](#_Toc307084130)

[1.3 Overview 2](#_Toc307084131)

[1.4 Definition and Acronyms 2](#_Toc307084132)

[2. System Overview 3](#_Toc307084133)

[2.1 System Evolution Description 3](#_Toc307084134)

[3. System Architecture 3](#_Toc307084135)

[3.1 Architectural Design 3](#_Toc307084136)

[3.2 Decomposition Description 3](#_Toc307084137)

[3.3 Design Rationale 3](#_Toc307084138)

[4. Data Design 4](#_Toc307084139)

[4.1 Data Description 4](#_Toc307084140)

[4.2 Data Dictionary 4](#_Toc307084141)

[5. Component Design 5](#_Toc307084142)

[5.1 Html Component 5](#_Toc307084143)

[5.2 JavaScript MWF Component 5](#_Toc307084144)

[5.3 JavaScript Communication Component 4](#_Toc307084145)

[5.4 PHP Component 4](#_Toc307084146)

[5.5 Database Component 4](#_Toc307084147)

[6. Human Interface Design 4](#_Toc307084148)

[6.1 Overview of User Interface 4](#_Toc307084149)

[6.2 Screen Images 5](#_Toc307084150)

[7. Requirements Matrix 13](#_Toc307084151)

# 1.0 Introduction

## 1.1 Purpose

This software design document describes the architecture and system design of Whoop-Txt. This document is intended for a technical audience and, in particular, for developers, project managers, and testers. Both high and low-level technical aspects for this project will be discussed in full detail and will focus on the functionality of each system layer.

## 1.2 Scope

Whoop-Txt is being developed to run on all major mobile devices and will integrate social networking and mobile platforms such as Google+, Facebook, and Google Maps to create an alternative method for text-sharing. Our goal is to introduce a user friendly mobile web application that offers features such as geo-location awareness and social network sharing.

## 1.3 Overview

This document will describe the architectural design of both the back-end and front-end interfaces for the Whoop-Txt software. Firstly, the functionality and modules required for the group creation/management, whoop sharing, and whoop-txt conversations listing will be described in full detail for the back-end interface. Then, the architectural design for the user-interfaces which include the authentication, message, menu, new message, and groups pages will be explained in more detail and screenshots and objects will be provided.

## 1.4 Definition and Acronyms

AJAX – Asynchronous JavaScript and XML

DOM – Document Object Model.

HTML DOM – defines a standard for accessing and manipulating HTML documents.

JSON – JavaScript Object Notation

MWF - Mobile Web Framework.

SQL – Structured Query Language; manages data in database systems.

# 2.0 System Overview

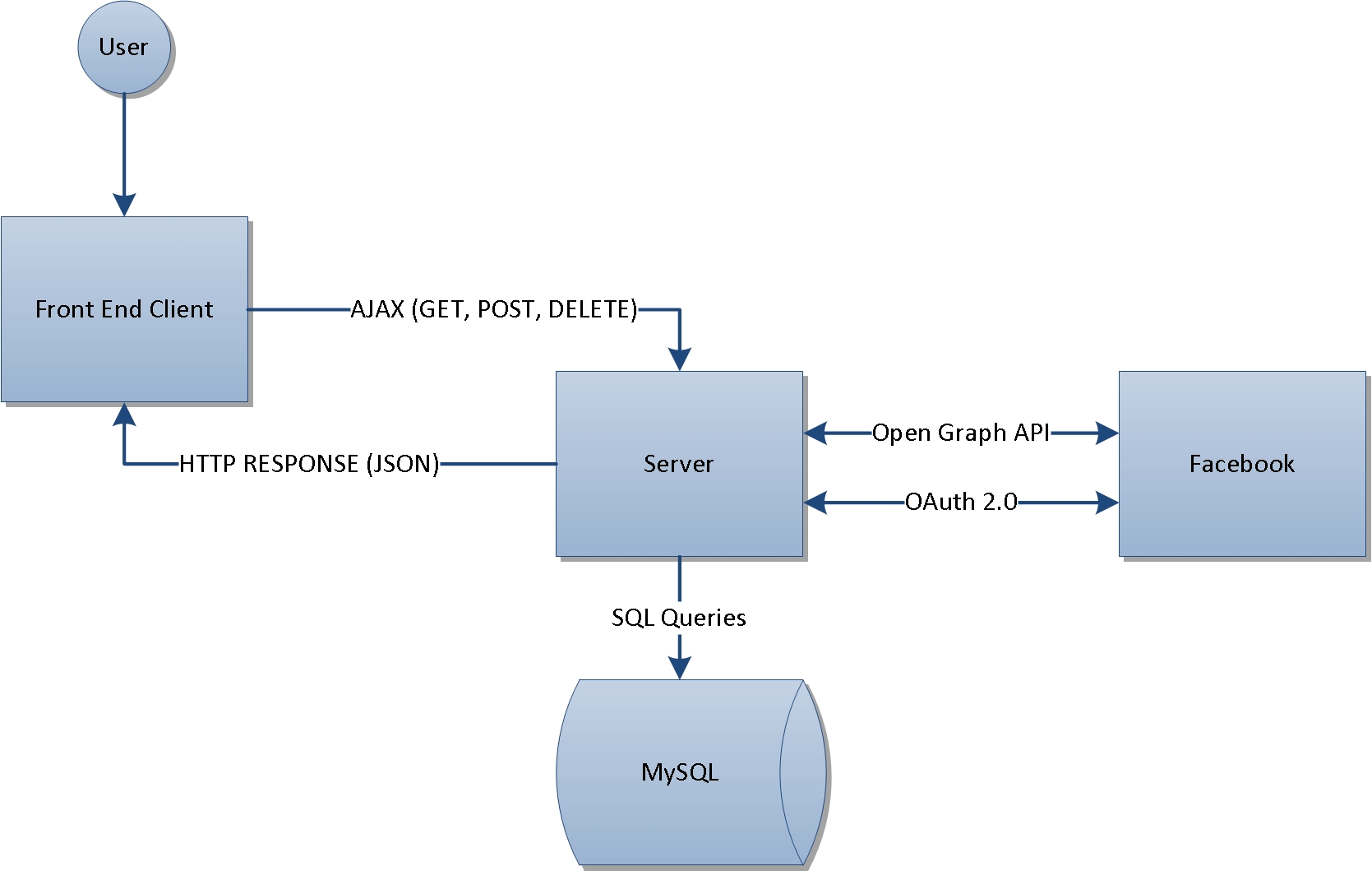
## 2.1 System Evolution Description

This project is being designed using an incremental approach. There are three primary stages to the design development including in Phase 0, due October 19, 2011; Phase 1, due the week of November 10, 2011; and Phase 2, due the week of November 28, 2011. Phase 0 consists of the creating a concrete design of the system and user interface and cementing all preliminaries needed to start implementation. Phase 1 will involve implementing the database as well as including log-in as well as online group creation fully functioning. Lastly, Phase 2 will be the final stage in which API’s will be fully incorporated and the final touches on the web-design will be made. The final product will be delivered to the customer the week of December 2, 2011.

# 3.0 System Architecture

## 3.1 Architectural Design

Please see below architecture design for the application:



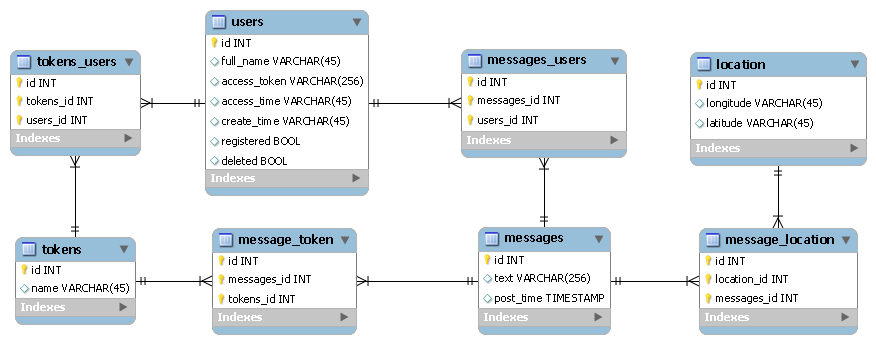
## 3.2 Decomposition Description

Users are either mobile users or desktop users that will be logged into Whoop-Txt. The front end client utilizing JavaScript will issue AJAX requests to the server which after committing appropriate queries will return query results in JSON format wrapped in an HTTP response. In order to authenticate and authorize users, the server will contact Facebook and utilizing OAuth 2.0 protocol will gain a privileged access token to be used for future requests. Furthermore,

## 3.3 Design Rationale

An important consideration with the system architecture was decupling front end design from the server side design. Using technologies such as AJAX/JSON it is possible to separate responsibilities for the server side developer and the front end developer. Furthermore, if in the future the team decides to design a native application it would be much easier to integrate the current decoupled structure into a native environment.

# 4.0 Data Design



The above figure represents the database schema for the application. A user is uniquely identified by his/her ID. From the users table we can go further to retrieve stored log in information and group information (hence the ‘tokens’ tables) and what messages are associated to a specific user.

The ‘messages\_users’ table associates a text body, a time, and a location to a message. This way, when a user is accessing the app, they can submit their current location to the app as a search parameter to what location based messages are pertinent to them.

Queries to this database will be fairly basic. They will consist of inserts to the users table upon a registration or invitation to the application, and inserts of new messages. The next main interaction will be with queries for messages that need to be sent to users. In the case of location messaging, the app will query based on the users current longitude and latitude coordinates and a span of time (say within the last 5 minutes).

## 4.1 Data Description

|  |  |
| --- | --- |
| **Users** | Users table stores general information about the user including last access time, and full name. New fields will be added throughout the development to store data retrieved from Facebook and other social sites. An interesting element is access\_token field–this field stores the OAuth 2.0 access token retrieved from Facebook that will later allow accessing more information about the user. |
| **Messages** | The messages table stores text messages sent to by users. Coupling tables will associate a sent message to a location, users, and groups/tokens. Post time field will be used to determine if a message is still valid to be sent to nearby users. |
| **Tokens** | Tokens are like categories but can be used like groups. For example, “Roommates” could a valid token that would group certain users. |
| **Locations** | Location represents GPS longitude and latitude. It’s signifies the location where the message originated from. |

## 

## 4.2 Data Dictionary

|  |  |
| --- | --- |
| **Users to Tokens** | Each Token can be matched to a list of users. A token is the primary means by which a user is associated to a group. |
| **Users to Message** | Each message corresponds to the user who authored it. |
| **Messages to Location** | Each message corresponds to a set of Longitude and Latitude coordinates that signify its geographical point of origin. This way, a user can use their current location to query the database for messages that were sent within their immediate vicinity recently. |
| **Messages to Tokens** | Messages are mapped to users, who are in turn mapped to tokens. By a query, it can be determined who receives a message by inputting the id of a token for a group, and finding which users are mapped to the same token. These users will end up receiving the message. Tokens however cannot be linked directly to specific messages. |

# 5.0 Component Design

The components of the Whoop-Txt system will be broken down according to the frameworks they interact with and the functionality they provide. This will allow for a strong separation of concerns and an easy to navigate code base. These main components will include a HTML front end, a JavaScript utility component for generation of Mobile Web Framework Code based on data received from the back end, a JavaScript utility component for communication with the back end PHP code, a PHP back end component for interfacing with the database, and a database component for the storage of user data and messages.

Creating these components in this way allows for a layered structure with a clear separation of both functionality and the languages being used in the different components. Additionally in this structure, a component interfaces with at most two other components, the higher level component and lower level component, creating architecture with low coupling.

## 5.1 HTML Component

The HTML component will be the framework for each individual page to be generated. This is the front end of the application and does most of the page creation and styling. This component will heavily utilize the UCLA Mobile Web Framework for generation of mobile friendly pages for all devices. This framework will be filled in with actual data such as groups, invitations, and messages on certain pages by JavaScript MWF Component. All parts that do not change based on data in the database will be generated here, while all dynamic data will be filled in later as a responsibility of the lower level component.

## 5.2 JavaScript MWF Component

The JavaScript MWF Component will be responsible for generating MWF code to display dynamic data such as a user's messages, groups, or invitations. This component will be the bridge between the JavaScript Communication Component that will get data from the back end and the HTML component that will be a static display written in HTML code. This component will be receiving data in a JSON format and converting that data appropriately to HTML DOM elements that will be placed on the users screen. This component will have an interface of JavaScript functions that can be called by the HTML Component to generate the data appropriate for the current page.

## 5.3 JavaScript Communication Component

The JavaScript Communication Component is the bridge between JavaScript front end code and PHP back end code. It is responsible for doing AJAX requests to the PHP code to retrieve data from the database. This is the last component that is executed on the client side and it interfaces with the server side code. Requests will be sent using a standard AJAX format and responses will be sent in JSON. This JSON response will then be send up a level to the JavaScript MWF Component for generation HTML code. This component will have a JavaScript interface that the JavaScript MWF Component can call to get data from the back end.

## 5.4 PHP Component

The PHP Component will be the first level of server side code creating a bridge between the database and the client side code. The PHP Component will be written in PHP utilizing SQL to communicate with the database. All updates or requests with the database will have to pass through this component so it will be important that this component is very secure and safe from malicious attack. The PHP Component will respond to AJAX requests by converting responses from SQL queries into JSON formatted results. This code will provide an interface in the form of different URLs that take specified parameters and give appropriate JSON responses.

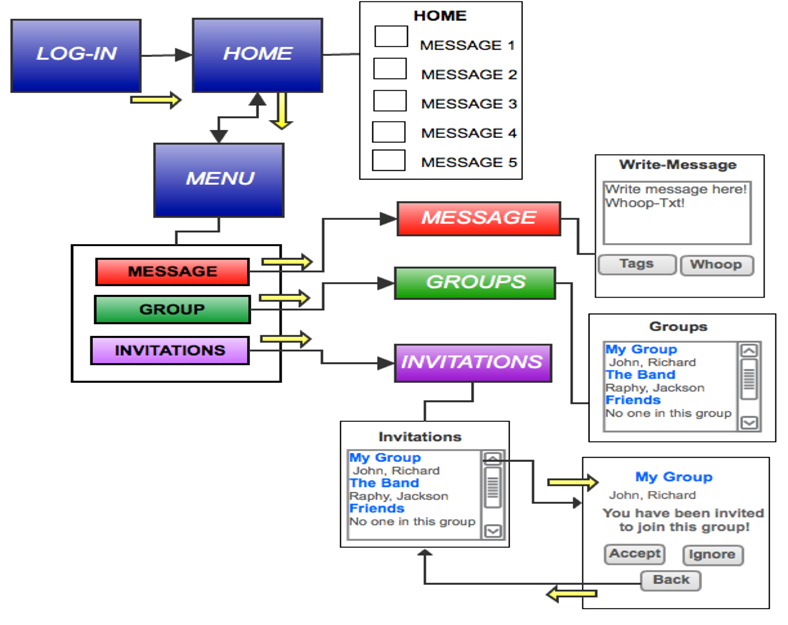
## 5.5 Database Component

The Database Component will be responsible for storing all of the data for the application. Access may be given to authenticated users via SQL queries. It is integral that this component is efficient, scalable, secure, and fault resistant since it will receive a large amount of requests and must maintain the integrity of all of the application data. The Database Component is further broken down into different tables for the different types of data stored for the application. This can be referenced in section 4 on Data.

# 6.0 Human Interface Design

## 6.1 Overview of User Interface

The main authentication page on the Whoop-Txt website is where users log in using their Facebook account. If the user does not have a Facebook account, our website will have the capability for a user to create a Facebook username and password by having a create account pop-up message on the main authentication page. After logging in, the user is navigated to the Home page which is a message stream page where Whoop-Txt conversations will be listed in most-recent order. The user can either scroll/read through the conversations listed or can navigate to the Menu page by pressing the Menu button located at the top of the screen. Once at the Menu page, the user can be directed to three different web pages: Home, New Message, or Groups. At the New Message page, the user will be able to texts to their Whoop-Txt groups, tagged individuals, and/or Whoop-Txt users within a locational range. The Groups page lists the groups that the user has either been added to, or has created. Here, the user will have the option to remove themselves from a group, add more friends to an existing group, or to create a new group. A high-level view of these Whoop-Txt pages is as follows:



6.2 Screen Images

|  |  |
| --- | --- |
| **Authentication Page**  The user will be able to log into the application via Facebook’s login page. |  |
| **Home Page**  Home page will list the most pertinent and up-to-date messages to the user. | Macintosh HD:Users:anthonybalmeo:Desktop:Screen Shot 2011-10-22 at 2.24.52 AM.png |
| **Menu Page**  Although currently the menu page lists three options, the team expects the number of menu items in this page to grow substantially. | Macintosh HD:Users:anthonybalmeo:Desktop:Screen Shot 2011-10-22 at 2.24.40 AM.png |
| **Invitations Page**  From this page, the user will be able to view all the pending invitation. | Macintosh HD:Users:anthonybalmeo:Desktop:Screen Shot 2011-10-22 at 2.25.24 AM.png |
| **Invitation Ignore/Accept Page**  The user will either be able to accept/or reject a given invitation. | **Macintosh HD:Users:anthonybalmeo:Desktop:Screen Shot 2011-10-22 at 2.25.34 AM.png** |
| **Groups Page**  The Groups Page will list all the groups available to the user. | Macintosh HD:Users:anthonybalmeo:Desktop:Screen Shot 2011-10-22 at 2.25.07 AM.png |

# 7. 0 Requirements Matrix

|  |  |  |  |
| --- | --- | --- | --- |
| **Req.#** | **Name** | **Description** | **SDD Sec. #** |
| 3.1.2a | Group Creation and Management | Create groups |  |
| 3.1.2b | Group Creation and Management | Invite friends |  |
| 3.1.2c | Group Creation and Management | Accept/decline invites |  |
| 3.1.2d | Group Creation and Management | Unsubscribe from group |  |
| 3.2.2a | Whoop Sharing | Select tags to share posts |  |
| 3.2.2b | Whoop Sharing | Select groups to share posts |  |
| 3.2.2c | Whoop Sharing | “Public” tag option |  |
| 3.2.2d | Whoop Sharing | “Nearby” tag option |  |
| 3.3.2a | Whoop-Txt Conversations Listing | View list of available posts |  |
| 3.3.2b | Whoop-Txt Conversations Listing | Select post and view content and members |  |
| 3.3.2c | Whoop-Txt Conversations Listing | Share a reply to a post |  |
| 3.3.2d | Whoop-Txt Conversations Listing | Re-share posts |  |
| 4.1.1 | User Interface | Facebook log-in | 6.2 |
| 4.1.2 | User Interface | Home page contains message stream | 6.2, 6.3 |
| 4.1.3 | User Interface | Menu page three navigation buttons: Home, New Message, Groups | 6.2, 6.3 |
| 4.1.4 | User Interface | New message page: write texts and tag | 6.2, 6.3 |
| 4.1.5 | User Interface | Groups page: lists subscribed groups | 6.2, 6.3 |
| 4.3 | Software Interfaces | MySQL, PHP, Apache, and Facebook application management software |  |
| 4.4 | Communication Interfaces | Management console based on HTTP protocol running on an Apache server |  |
| 5.3 | Security Requirements | SSL Encryption |  |
| 5.4 | Software Quality Attributes | Error Handling |  |