OULU UNIVERSITY OF
APPLIED SCIENCES

Anselm Tochukwu Ogbunugafor

Developing a Bulk SMS Web Application with Integration to a Social Network

Developing a Bulk SMS Web Application with Integration to a Social Network

Anselm Ogbunugafor
Bachelor's Thesis
Autumn 2012
Oulu University of Applied Sciences
Degree Programme in Information Technology

PREFACE

This project was started and completed by me, Anselm Ogbunugafor. The supervisor was Veikko Tapaninen, He helped with guidance and constructive criticism. There was no client representative since this was a private project.

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Information Technology

Author: Anselm Ogbunugafor
Title of thesis: Developing a Bulk SMS Web Application with Integration to a Social Network
Supervisor: Veikko Tapaninen
Term and year of completion: Autumn 2012 Number of pages: 50

Sending a bulk SMS with a mobile phone is often a task that is fraught with usability issues. It becomes evident that a mobile phone with its small form factor cannot deliver the level of user experience that a web application can.

This project aims to develop a web application where users can register, create and manage contacts and contact groups, send bulk SMS. Facebook will be integrated into the web application to enable users to sign up and login to the service. A Facebook page will also be created for the service where the web application can be used from there.

In developing the application, PHP, HTML, CSS and Javascript will be the tools used to accomplish this.

This project can further be extended in the future to allow 2-way SMS campaigns. It can also be used as a service to support critical operations like ambulance service, rescue and any operation that might need short automatic message distribution when an event occurs.

Keywords: SMS, PHP, Social Networks, LAMP Stack, Facebook

TIIVISTELMÄ

Oulun seudun ammattikorkeakoulu

Tietotekniikan koulutusohjelma

Kirjoittaja: Anselm Ogbunugafor

Opinnäytetyön nimi: Irallisten viestien web-ohjelman rakentaminen Facebook Integraatiolla.

Valvoja: Veikko Tapaninen

Termi ja valmistumisvuosi: syksyllä 2012 Sivumäärä: 50

Irtonaisien viestin lähettäminen puhelimella on usein tehtävä, joka on täynnä käytettävyysongelmia. On ilmeistä, että matkapuhelin pienellä käyttöjärjestelmällään ei kykene toimittamaan samaa käyttökokemuksen tasoa, johon web-sovellus pystyy.

Hankkeen tavoitteena on kehittää web-sovellus, jossa käyttäjät voivat rekisteröityä, luoda ja hallita yhteystietoja ja yhteysryhmiä, lähettää irtonaisia viestejä. Facebook liitetään web-sovellukseen mahdollistamaan käyttäjien kirjautumisen ja rekisteröitymisen palveluun. Facebook-sivu luodaan myös palveluun, jota kautta web-sovellusta voi myös käyttää.

Tätä sovellusta kehittäessä, PHP, HTML, CSS ja JavaScript ovat työkaluja tämän saavuttamiseksi.

Hanke voidaan edelleen laajentaa tulevaisuudessa sallimaan 2-suuntaisia SMS kampanjoita. Sitä voidaan myös käyttää tukipalveluna kriittisille toiminnoille, kuten ambulanssipalveluille, pelastukseen ja mihin tahansa toimenpiteeseen, joka saattaa tarvita lyhyitä automaattisia viestejä tarvittaessa.

Avainsana: SMS, PHP, Social Networks, LAMP Stack, Facebook

TABLE OF CONTENTS

1 INTRODUCTION

Web applications are fast becoming ubiquitous. There has been a paradigm shift from the platform dependent desktop applications in recent years. The ease at which web applications are deployed and maintained have to a large extent contributed to their dominance and ubiquity. This is a view taken from the organisation/company standpoint. When looking from the user's perspective; the ease of firing up a web browser and visiting an address where he can find out when the next bus arrives at his bus stop is also appealing and effortless. It is a win-win situation for both the organisation and the end-users.

The proliferation of web browsers on all platforms, mobile and embedded devices has made this possible. This has given a whole new meaning to the "Write Once Run Anywhere" slogan. It has also made it less enticing to continue to develop and maintain the desktop applications of different flavours for different platforms. Why go through the pain of writing a journey planner application like www.reittiopas.fi, creating different versions of it for a Windows machine, Linux machine, Mac machine. Come up with a Software update mechanism for each of the platforms, where end-users will continually and manually have to update the application on their respective machine. Have different user experience on each of the platforms since a complete user-interface replica cannot be achieved on the different platforms. You see all this is what a web application magically solves. The benefits of developing a web application against a desktop application are listed below:

- Single codebase will run on all platforms which have a web browser.
- Same user-experience across all browsers on all platforms.
- Centralised code maintenance. Do the updates on the codebase and all users are updated at once.
- End-users do not have to do anything to keep the application up to date.

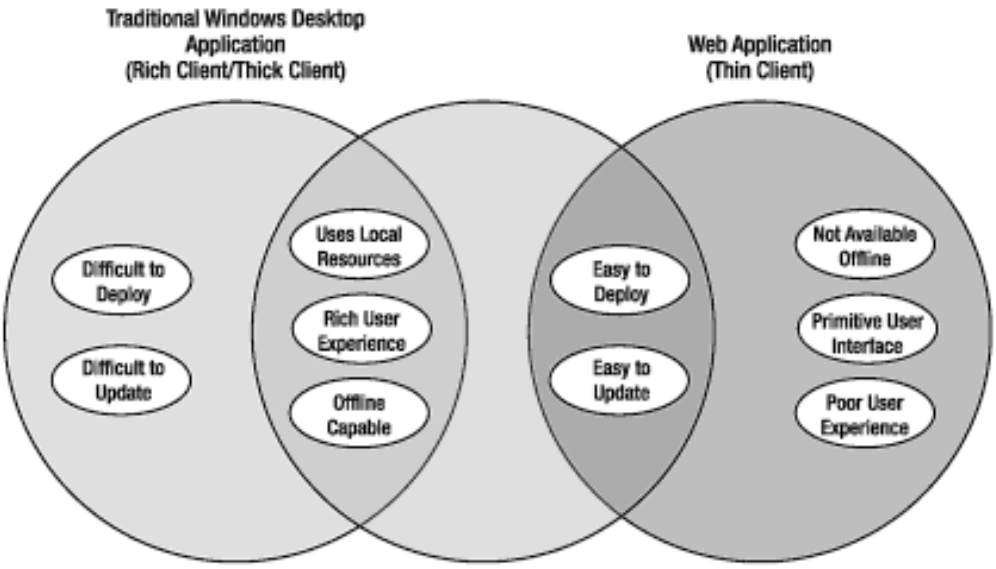


Figure 1. Deploying .NET Applications. (Sayed & Sayed 2006, 3).

No hassles maintaining different codebase for different platforms. One single codebase to rule them all, but of course web applications are not the silver bullet to solve all our computing problems. Browsers are still trying to catch up with all the technologies that are available for a desktop application. Web technologies have not matured to the point of creating specialised applications like 'Auto CAD' for the web. With the rapid rate of changes in Information Technology, this may no longer be the case in five years time. We can look at "Google Docs" (a Word and Spreadsheet Processor application built by Google), which is solely a web application and is gaining wide usage as a collaborative suite. Also, with HTML5 specification, which solves some of the complexities of embedding rich documents and introduces new offline storage mechanism, the web application space is bound to be the de-facto application model.

1.1 Thesis Project Objectives

On this project, we set out to develop a web application for sending out Bulk SMS (Short Message Service) with an additional feature of contact management. We will then integrate the application into a popular social networking site: Facebook.

I chose this project because of the interesting challenges it poses. This project tries to solve the limitations of using a mobile phone to send out SMS in bulk. The user experience of doing this on a mobile phone is appalling. This project sets out to fix this problem by implementing its usage on

a web browser, thereby improving the ease at which SMS is sent out in bulk, the group contact management, the visibility of delivery reports and the airtime credit usage. While doing it, we throw in some social network integration to drive up user engagement on the app. This project will lay the foundation for future implementation of Short Code Messaging, which is, used in lotteries, raffle draws and SMS campaigns. At the end of this project, we should have learnt and acquired the technical skills necessary to accomplish a web application software project. That is the main objective of this project

1.2 Web Application Stack

Developing a web application involves combining different technology stacks in a cohesive manner to produce the end result. This stack is comprised mainly of three parts:

- Front-End / Presentation layer uses HTML, Javascript and CSS
- Backend / Business Logic layer uses scripting languages like PHP, Python, Perl
- Data Storage layer uses storage engines like MySQL, Postgres, Oracle, MongoDB

Each of these stacks are implemented using different technologies and there are a lot of varieties to choose from when building your next web application. We will delve deeper into these in later chapters.

1.2 Social Networks

Almost all of our physical interactions with the real world have been re-created and mapped onto the web/Internet. The web has become more of a walled garden where all basic services and interactions with the outside world can be achieved. We now run our banking transactions, read the latest news and mails and order dinner on the web.

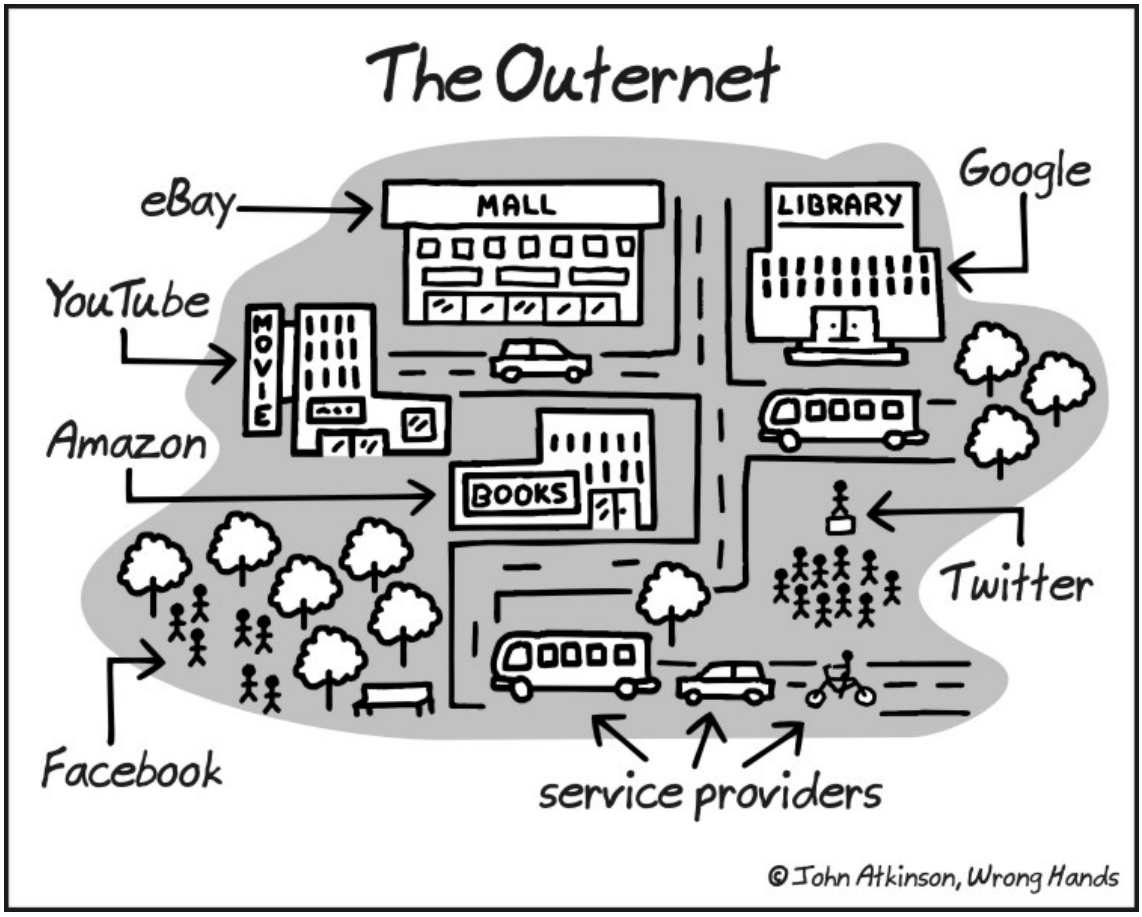


Figure 2. The Web as a walled garden. (Atkinson, 2012. Date of retrieval 4.5.2012.)

Social networks are the latest addition to the web services. Once privacy concerns had been addressed and users felt at ease sharing their private lives on the web, it was not long before a port of our social interactions appeared on the web. Social Network sites started being available on the web as early as 1997.

SixDegrees.com was the first to combine some social networking features together to form a social site. (Wikipedia, 2005, Date of Retrieval 12.1.2012.) Basically, a social networking site allows users to create public profiles of themselves, discover other profiles by which they are linked, create a friend/connection list, allow messaging between profiles, share titbits of private information across profiles like pictures, links, reviews, recommendations, location, status updates.

There are different social sites on the web catering to different categories, communities and interests. Examples of these categories are professional, business, classmates, corporate,

academic, ecological and musical networks. Some of the most popular social networking sites are Facebook, Google+, LinkedIn, Twitter, Orkut and MySpace.

Facebook is important to us because it is one of the biggest social networks with a user base of hundreds of millions. Facebook tends to a niche of friends and acquaintances that is of interest to us. Additionally, Facebook allows third-party developers to build applications like games, chart travel histories, birthday calendars. This particular feature has made users on Facebook to be more engaged and spend more time on the site. It has skyrocketed Facebook to be one of the most visited sites. We are very much interested in this feature because it allows us to integrate the SMS application to Facebook. By doing so, our users can access their contact list directory from which they can send their bulk SMS. There would be no need for duplicating their contact lists. Also, since the user engagement is high on Facebook, the users will not need to login to the SMS application to send an SMS to their friends and colleagues. The benefits of integrating to Facebook are listed below:

- Single Sign on.
- Single Profile.
- Easy access to contact list.
- Increased user engagement.

2 DEVELOPMENT DESIGN AND PLAN

The aim of the design and planning phase is to be able to set out the parameters to what is achievable, the technological stack to be used, what show stoppers we would anticipate and how to work around them. The strengths and limitations of any implementation are weighted at this stage of work. At this stage, we set out our requirements for the application and based on these requirements, we choose a solution that is malleable and scalable.

Malleable in the sense that as the application is being developed or used, there might arise more use cases. We would like to incorporate these additional use cases in our current solution without the need of tearing down the current solution and starting again from ground zero. Any solution that can easily absorb new requirements/use cases in an efficient manner is what we aim at.

Scalable in the sense that, once the application is deployed, our user base will grow. We want to create a solution that also grows with our needs, one that scales up easily when our user base grows and scales down when our user base shrinks. A solution that we can throw hardware to and it responds efficiently without making a complex architecture out of it.

2.1 The Requirements for the Service

Before any application is developed, the requirements that the application must meet are mapped out. The requirements are most often divided into two categories for the sake of clarity and simplicity. The categories are: Functional Requirements and Non-Functional Requirements.

Functional Requirements states out what the system should accomplish while Non-Functional Requirements states out how the functional requirement should be met and any road blockers that should be taken into consideration before the functional requirements can be met.

In layman's terms: Functional requirements state the business requirements that need to be met while Non-Functional requirements state the technical implementation that is used to meet the business requirements.

2.1.1 Functional Requirements

The functional requirements for the application are listed below:

- The user should be able to register an account for himself.
- The user should be able to place an order for the purchase of credits to his account.
- The user should be able to use the credits in his account to send SMS to any supported mobile phone numbers.
- The user should be able to create and manage several contact lists.
- The user should be able to select any of his contact lists for mass SMS sending.
- The user should be able to use his profile on a social network site like facebook.
- From the social network site, he should be able to perform the basic functionality of sending a SMS.
- The user should be able to reset his password when it is forgotten.
- An admin user should be able to view reports of usage by users or periods.
- An admin user should be able to change configured SMS Gateway routes.

2.1.2 Non-Functional Requirements

The Non-Functional requirements for the application are listed below:

- The user should be able to access his account at any time of the day.
- The application should be online and accessible by a browser.
- The UI design should be simple, logical and useable by the user.
- There should different user privilege roles.
- The application should be malleable
- The application should be scalable

2.2 System Architecture

For every deployed application, there is an architecture upon which it is built. Early on in our design, we chose the LAMP (Linux, Apache, MySQL, Perl/Python/ (PHP in our case)) stack as our technology architecture.

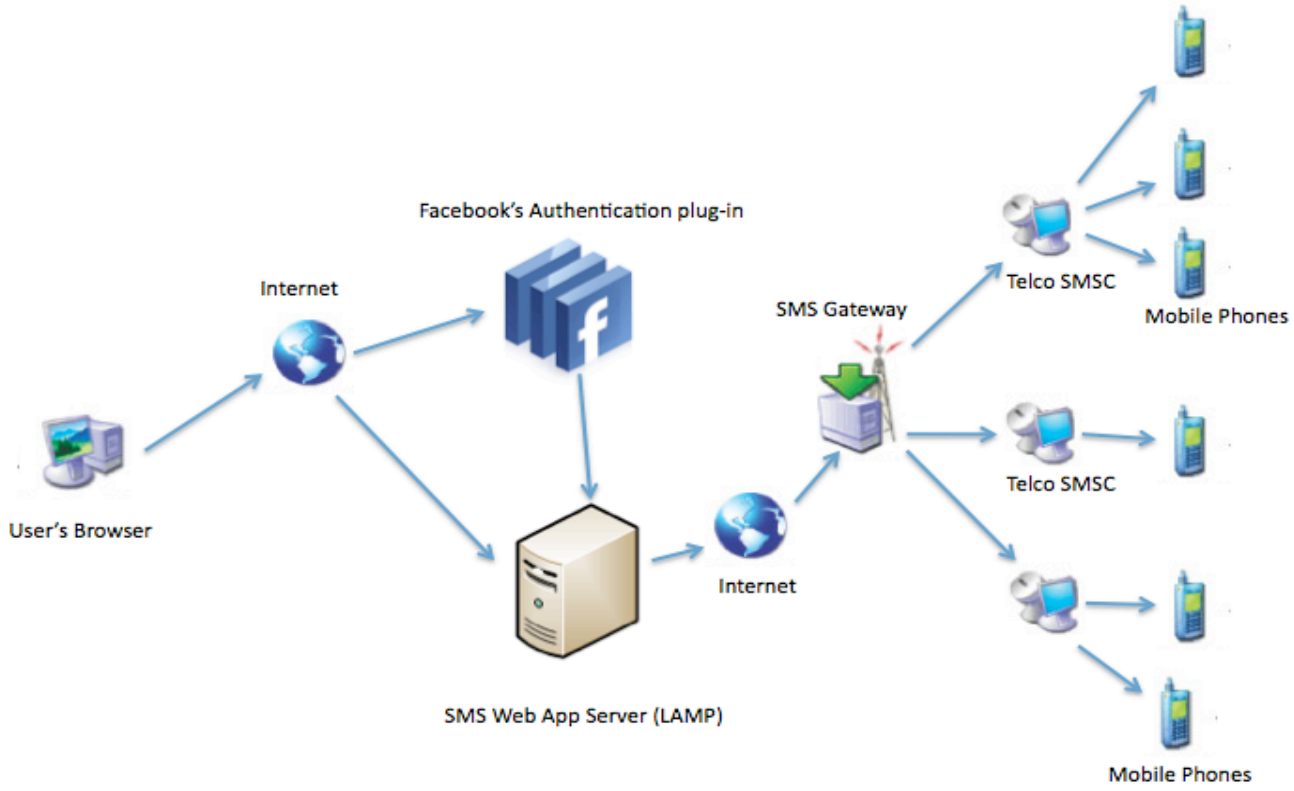


Figure 3. System Architectural Diagram

The LAMP stack is a popular stack in web development. All the parts that make up this stack are open-sourced, license free and have a vibrant support community built around them. This frees us from proprietary solutions like the Microsoft .NET platform and their per-seat/per-server license hassles (Hope & Walther, 2009, 12).

Linux is an operating system which derives its root from Unix. Therefore Linux is a Unix variant. According to w3techs.com, Unix holds a dominant 63% of the operating systems powering the web as at March 2012. The second runner up Windows, comes in at a measly 36% share. The Linux operating system powers most of the major and popular sites like Google, Amazon, Yahoo, Facebook etc.

Apache is a web server for serving up web pages. It runs as a process on an operating system, which in most cases is Linux. It also runs on Windows and on a Mac operating system. Apache is

also a dominant web server in terms of market share. The chart below depicts the current state of web servers powering the web.

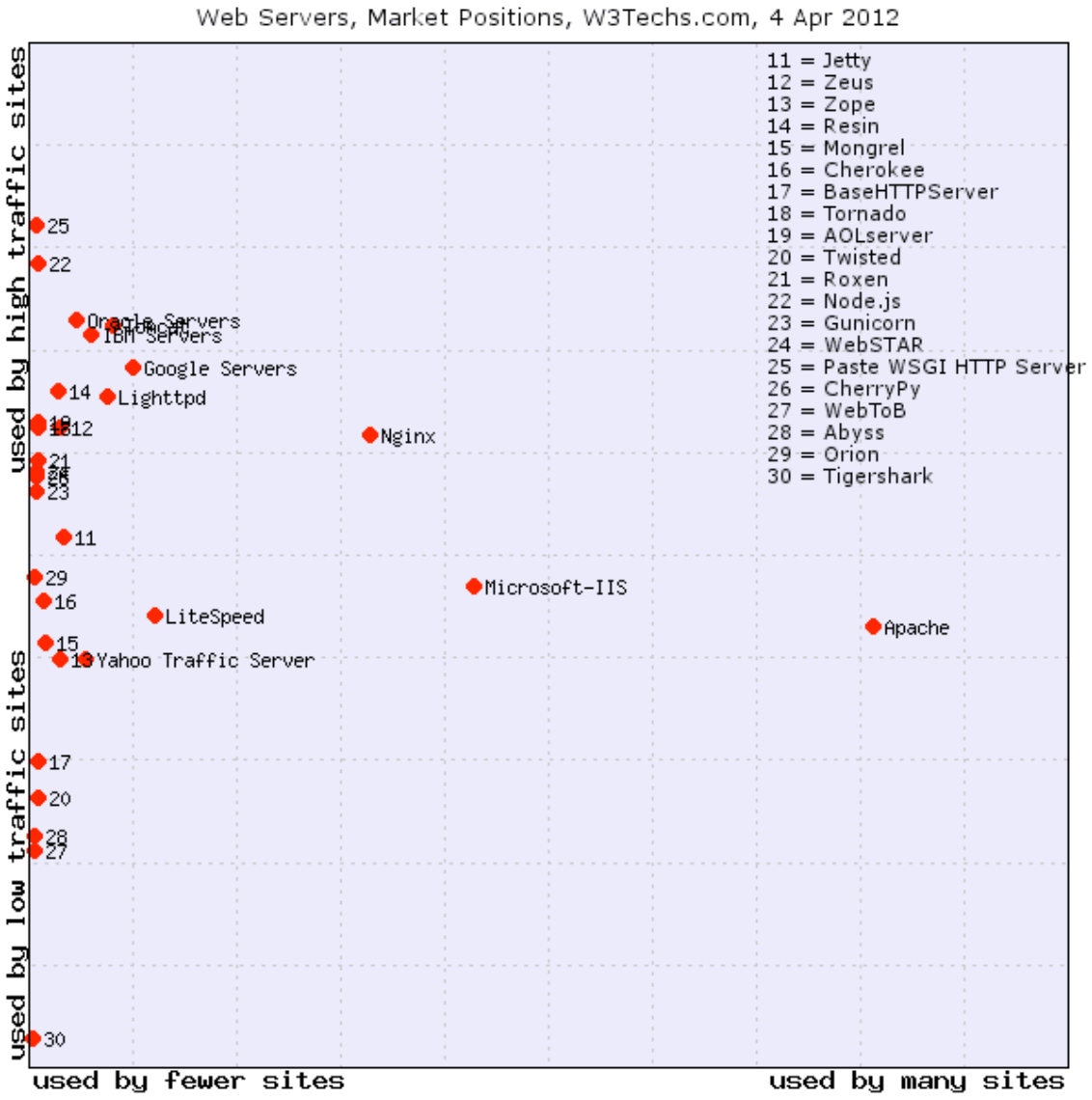


Figure 4. Market Position of Web Servers (Soltano, 2012, Date of retrieval 28.4.2012)

MySQL is a relational database storage engine used by any application that needs to store and retrieve data. MySQL is also a popular and fast database engine. Its MyISAM storage engine is one of the fastest, running millions of queries in a second. The other alternatives for a storage engine are proprietary Oracle Database and Microsoft's SQL Server.

PHP is a web scripting language, which in its early days (1995) was coined as 'Personal Home Page'. It is now known as 'PHP: Hypertext Processor', which is a recursive acronym. PHP is

widely popular as a web scripting language. The image below shows popular web scripting languages and their market share on the web. PHP is popular because of its simplicity, its' C++ syntax like and also a non-typed language. Its ease of adoption and low learning curve by new adopters of the language have led to its popularity.

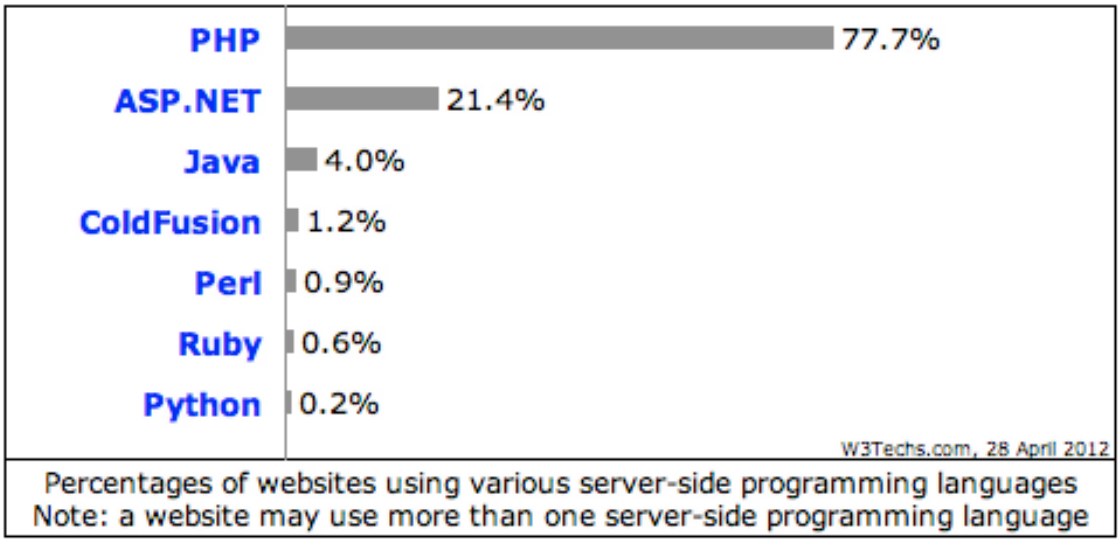


Figure 5. Market position of server-side languages. (Soltano, 2012, Date of retrieval 28.4.2012)

With this architecture it is easy to scale up quickly without necessarily making the architecture complex. When the user base of the application grows, scaling up can be done by:

- Having multiple instances of Apache running and moving a load balancer proxy in front of them.
- Using a memcache server for caching authenticated traffic and Varnish server for caching unauthenticated or anonymous traffic.
- Having multiple MySQL instances running. With a Master-Slave replication, we can have write operations piped to the Masters while normal read operations are piped to the Slaves.
- Using a MySQL table sharding technique to notch up performance.

2.3 Server Side Technologies

When it comes to web technologies, there are two major categories: Sever Side and Client Side. This follows the anatomy of having a web page delivered to you when you visit a link. Once you make a request for a web page (e.g. https://www.google.com/news) on your browser, the following transactions take place before you get served:

- A DNS (Domain Name System) request is made for the domain google.com. This helps to translate the domain google.com to a valid i.p. address (e.g. 173.194.32.39) and also provides a map of how to locate 173.194.32.39.
- Once the web server (e.g. Apache) residing on the google.com domain receives your request, it fires up the server side language (e.g. PHP) process to respond to the request 'news'.
- The server side language process receives the request for 'news', and runs the business logic for that request. It may have to make a call to the storage engine (e.g. MySQL) to retrieve data relevant to the 'news' request.
- Once the server side is done with generating a response for the request, it sends the HTML (Hyper Text Markup Language) response to the web server. HTML is the markup language understood by web browsers.
- The web server receives the response and sends it via the same route back to your machine using HTTP (Hyper Text Transfer Protocol) as the transport protocol.
- Your browser receives the response, parses it and then displays the information. In the HTML that your browser receives, there are references to other assets like images, videos, style sheets, and client code. Your browser makes an attempt to fetch all the referenced objects and loads them. If there is any client code loaded, it executes them.

Our server side technology of choice is the LAMP stack defined in the previous sub-heading.

2.4 Client Side Technologies

Client side technologies are executable codes that are ran on the browser to help with the rendering of the page and enrich the user interaction with application. They are said to be run at the client side.

There are different client side technologies but JavaScript is the most popular. Below is a chart of the existent client technologies and the web market share they hold.

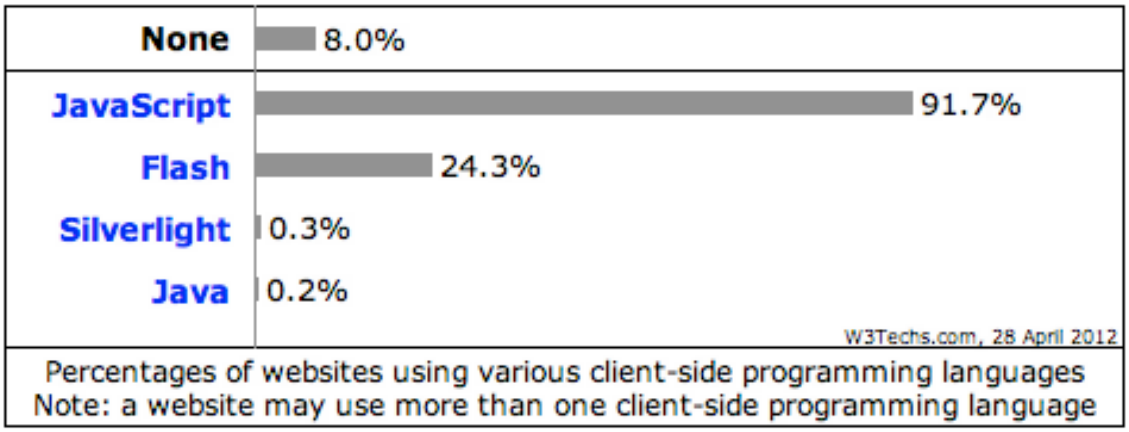


Figure 6. Market position of client side languages (Soltano, 2012, Date of retrieval 28.4.2012)

Our client side technology of choice is JavaScript.

2.5 Source Control and Versioning

In most professional programming works, be it open source or proprietary, there is always a need for Source Control. Source control tools allow a flexible collaboration within a team of developers. Source control helps in source code accounting, knowing the current state of work, what each programmer has checked in and who to blame for a code conflict or bug.

Source control helps to make sure that the team's effort are not overwritten, keeping version history of each document that was checked in, what was modified in each version. If there's a conflict it helps in resolving it. Source control also helps deployment teams in tagging a revision for deployment.

There are different source control tools. Some of the most popular are CVS (Concurrent Versions System), SVN (Subversion), VSS (Visual Source Safe), GIT, Bit Keeper, Monotone and many others. We are going to use SVN as our source control tool. I chose SVN because it supports atomic commits and has offline support. More so, I am more familiar with SVN. SVN will serve as a backup repository for our code and it will help with code reverts and tracking.

3 IMPLEMENTATION

At this stage we describe the processes that we have taken to develop and implement the application.

3.1 Development Environment

Producing software and writing code has to be done in an environment that is conducive and nurturing for the job at hand. A conducive environment for producing software is one where low level details are automated and taken care of so that the programmer does not incur any overhead with mundane tasks. An environment where the programmer can delve right into and start becoming productive from the onset is what we strive for.

A programmer's development environment will include things like a workstation, IDE (Integrated Development Environment), tools for remote access to servers, code repository, debugger, file transfer program, image manipulation and design application.

Our workstation of choice is the Apple MacBook Pro that runs the 10.6(Snow Leopard) version of Mac OS X. There are different IDEs that one can choose from for doing web development. An IDE has features for making coding easy and fun. Some of these features are code completion, syntax highlighter, easy code refactoring, debugger and so many others, depending on the IDE.
We could have used NetBeans, Eclipse, Visual Studio, ZendIDE, Notepad but we chose to use BBEdit which runs on a Mac. Its simplicity, speed and less bloat is what attracts us to BBEdit.

Our code repository resides in SVN. We use SSH (Secure Shell) for remote access to the web server and SCP (Secure Copy) for file transfers. We prefer the use of Firebug that runs on Firefox for the debugging of our JavaScript code. SQLEditor will serve as our database schema design tool.

3.2 Database Design and Schema

How we store and retrieve data from the database will give a huge impact on our application. Therefore, it is always a good idea to layout database and tables right from the onset. Since we

are using MySQL as our storage engine and MySQL is a Relational storage engine, we intend to use the primary keys, foreign keys feature to design a database that will have no redundant data in its tables. We also intend to follow strictly the 'Third Normal Form (3NF)' rule of database normalization. The Third normal form is a database normalization technique where the attribute columns, which are not dependent on the primary key, are moved to a separate table in order to take away any redundant data.

After the above conditions had been taken into account, we arrived at the schema plan shown below:

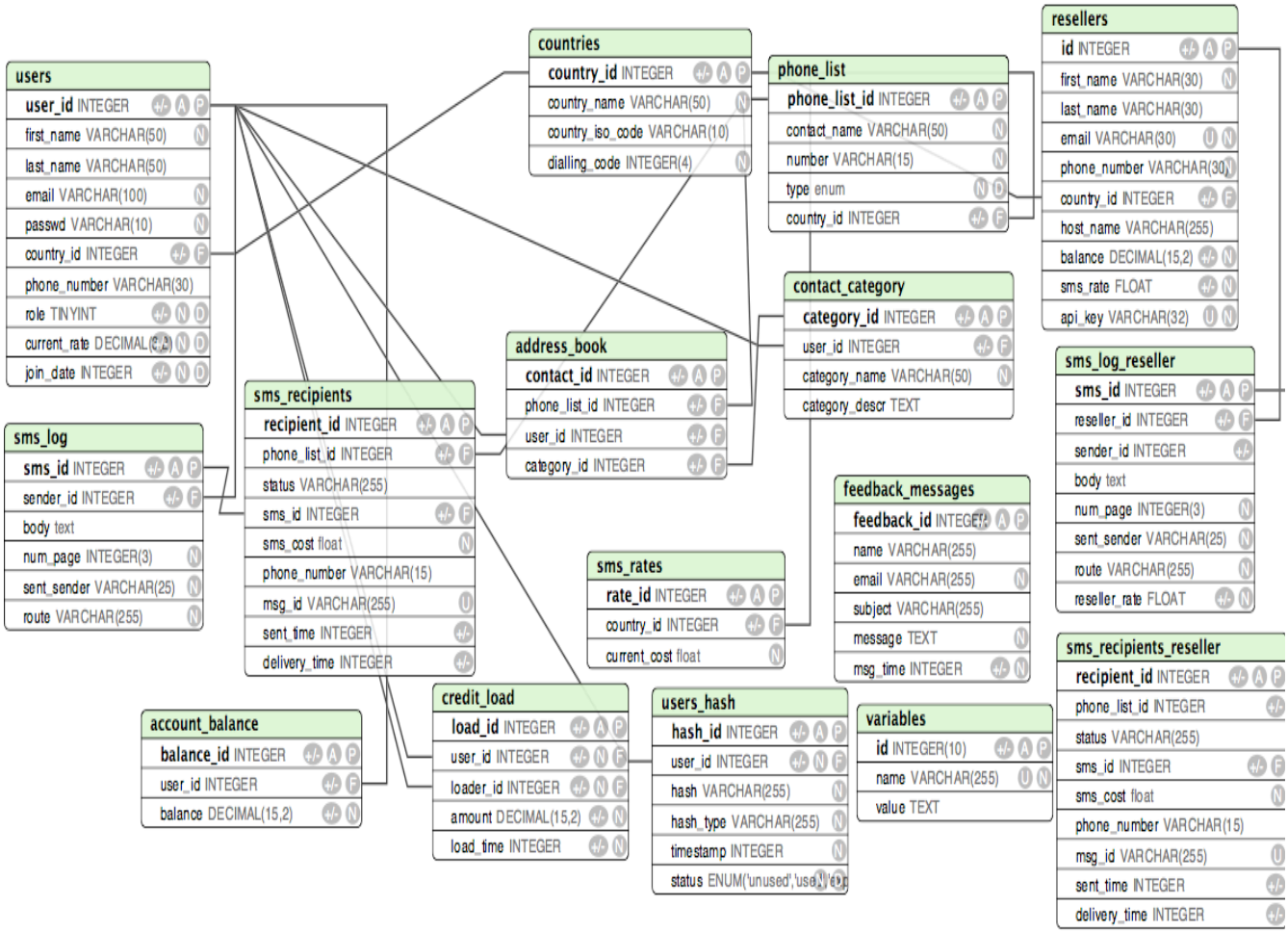


Figure 7. Database Schema Plan.

The schema plan above shows how data is structured in the app and the relationships that exist between the tables. In relational databases like MySQL, data are structured in tables. Database tables are a collection of rows (tuples). The rows are individual records, which in most cases are

unique. For each row, data is arranged in columns (attributes). The columns define how data is stored and the format (data type) in which the data is stored. The columns also show how the relationships across tables are mapped.

In the above schema, we make use of 16 tables to structure out the data. The tables contain information about the registered users, the bulk SMS they sent, their individual address book, the amount of credit in their account and other relevant details that are needed to smoothly run the app.

3.3 Code Igniter Framework

For the sake of productivity, code clarity, maintenance and best practices of the industry, we are not going to produce the application with vanilla, spaghetti PHP mixed with HTML. Spaghetti code is generally used to describe code that is unstructured. The program flow of spaghetti code is twisted and hard to keep track of. With spaghetti PHP, the PHP is mixed with HTML, JavaScript and CSS in a single file. Spaghetti code is generally hard to debug and has a high potential of introducing bugs. This is the reason I am going to adopt and use a framework that follows an MVC (Model View Controller) pattern.

The MVC pattern promotes code separation into logical units. This separation helps the team to focus independently on the part they are concerned with. They can develop, test and maintain each part separately. The MVC pattern separates code into 'Model', which handles the business rules, 'View' where the user interface is developed, 'Controller' which handles, translates and dispatches user input (Gamma, Helm, Johnson & Vlissides 1995, 4.)

In PHP world there are quite a lot of frameworks that follow the MVC pattern. Some of them are Code Igniter, Zend Framework, Symfony, CakePHP and Kohana. Our MVC framework of choice is Code Igniter. CodeIgniter contains clear and easy to understand code, favors configuration over code and is fast enough for our needs. CodeIgniter is an open source PHP framework licensed under the Apache/BSD (EllisLab, 2011, Date of Retrieval 12.3.2012).

3.4 JavaScript with the jQuery library

One major concern when writing client side script in JavaScript is the browser fragmentation problem. Any seasoned front-end web developer can describe you what a nightmare this problem is for them.

Different browsers have different implementations of their JavaScript engines and that is not all. Each of them honors the CSS (Cascading Style Sheets) web standard differently. The way JavaScript is interpreted and executed on Firefox is different from Internet Explorer just as it is different in Safari. The list goes on and on.

In order for our JavaScript code not to end up with browser sniffing snippets and different implementations for each of them, we are going to use a JavaScript library that does all the hard work for us. jQuery is a JavaScript library that helps us to achieve this aim. jQuery simplifies things like DOM(Document Object Model) traversing, event handling, ajax callbacks in a way that our JavaScript code would work in the same manner across all supported browsers.

3.5 Implementation Cycle

The active implementation and coding for this project started in earnest on 11 February 2011. I bought a VPS (Virtual Private Server) hosting package, installed SVN (Subversion). Apache, PHP and MySQL were already installed by the hosting company. Before this date, I had done some coding on my local machine. With my VPS and SVN setup, I could finally start the full development with SVN track logs.

The development came to a halt in August 2011, when I released the first stable version of the app. The development cycle took 6 months as a part time work. At the beginning of the project, I was devoting to it 15 hours every week but later cooled off to 5 hours a week towards the end of the cycle. In July 2012, I started implementing the Facebook integration feature. Generally this would continue to be a work in progress as there is always something to be improved or some new features to be implemented. The SVN log that details the project development can be found in the appendix section of this document.

Some screen shots of the web application developed are shown below:

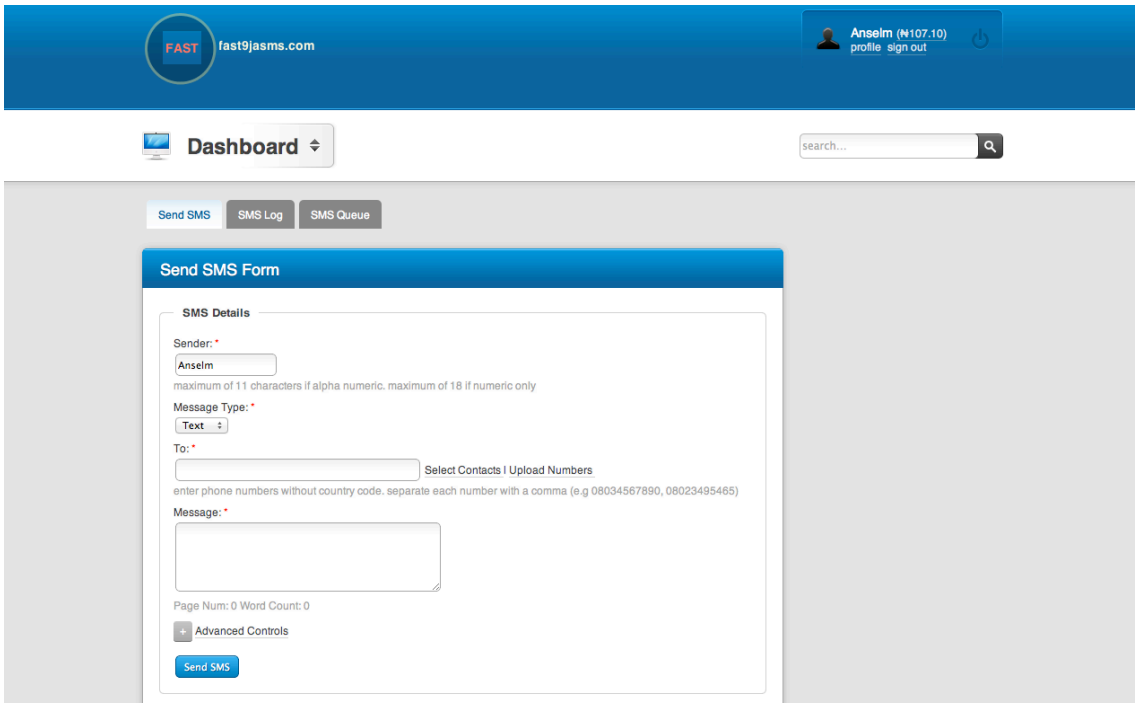


Figure 8. Bulk SMS sending page for a logged in user

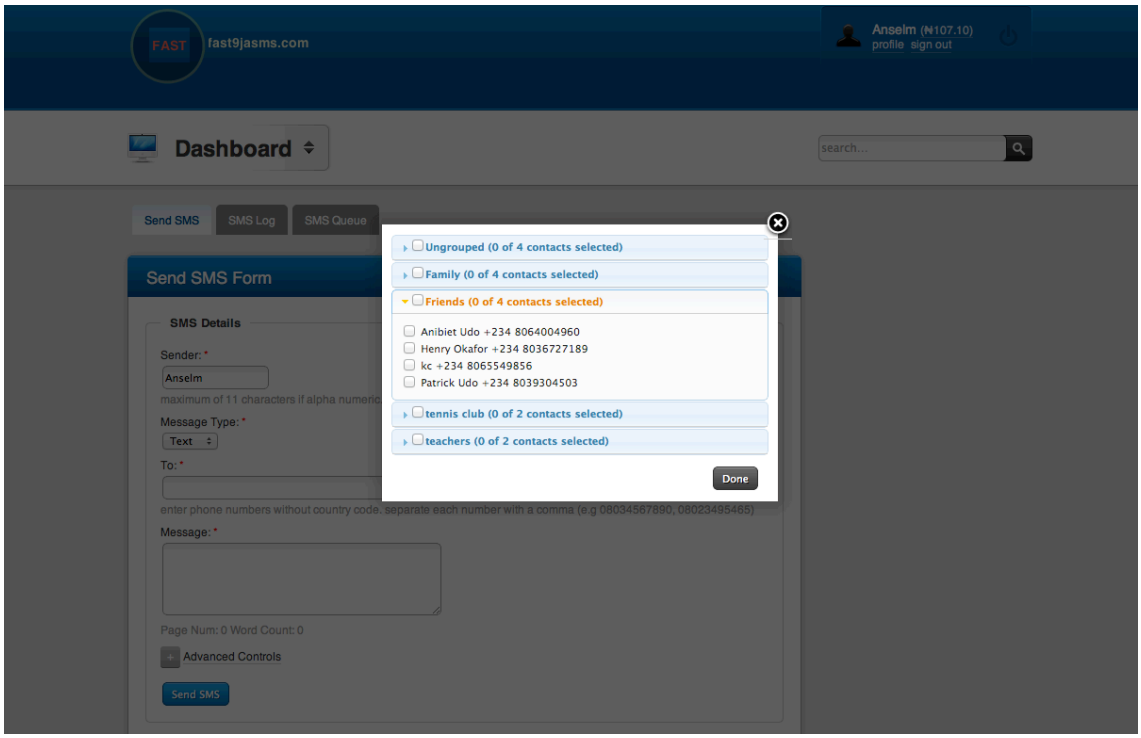


Figure 9. Contact Group selection for bulk messaging

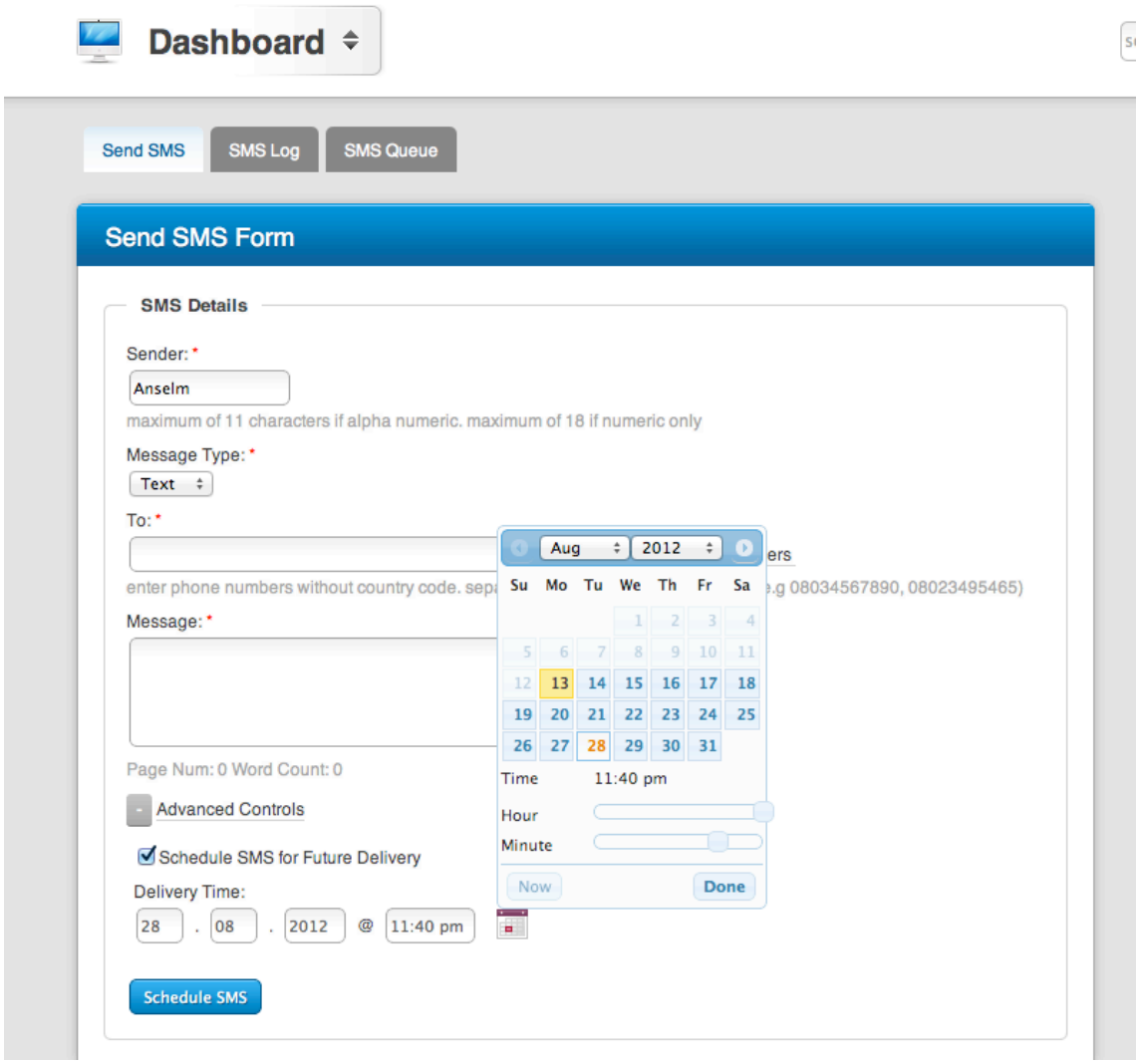


Figure 10. Scheduling future delivery of bulk SMS

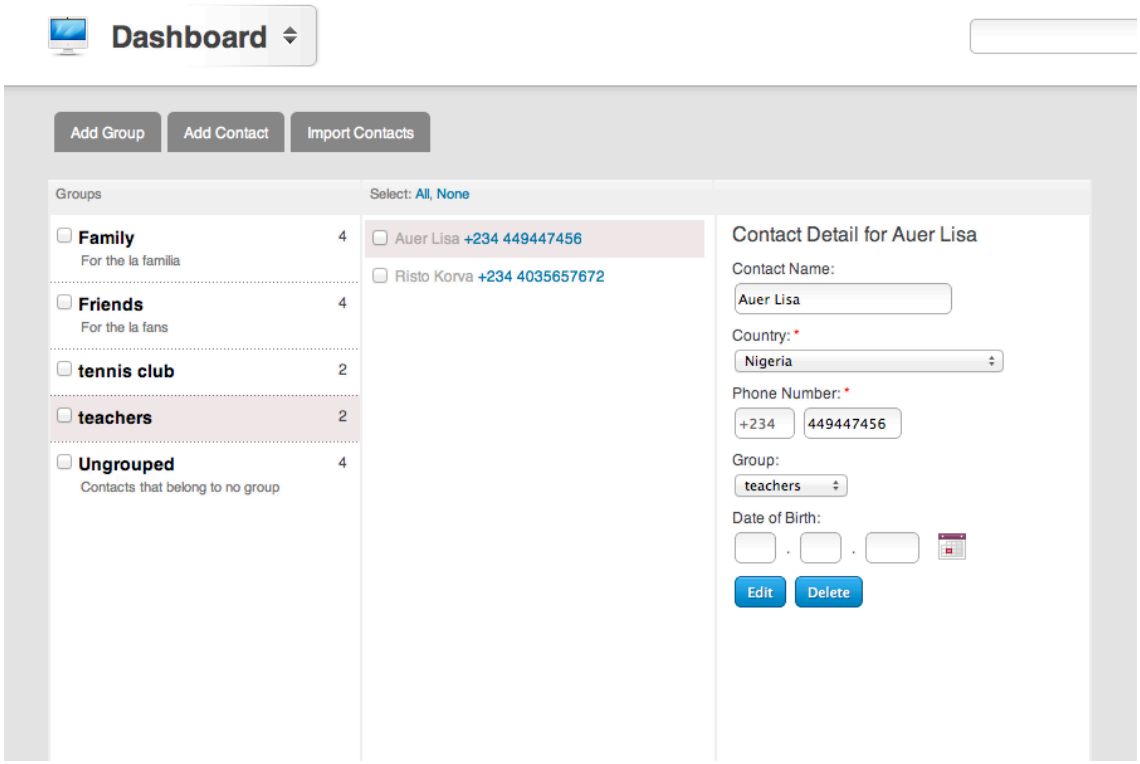


Figure 11. Contacts Management page

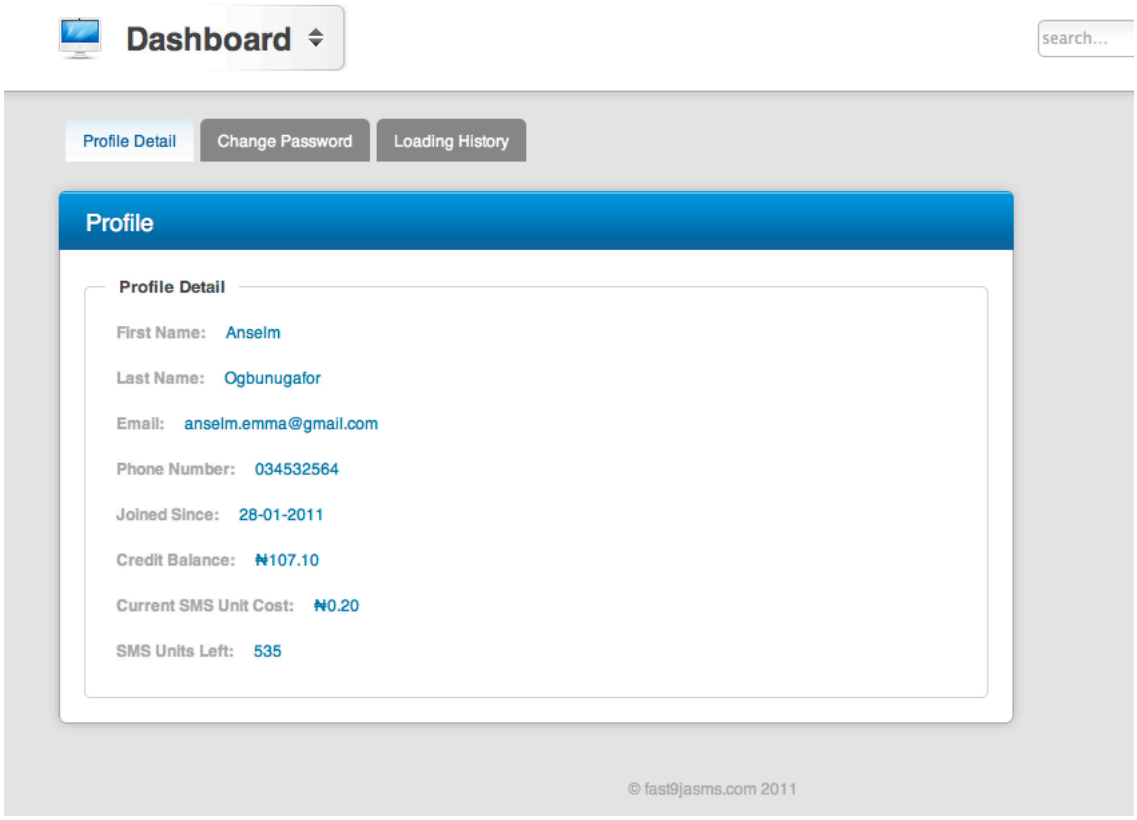


Figure 12. Profile page

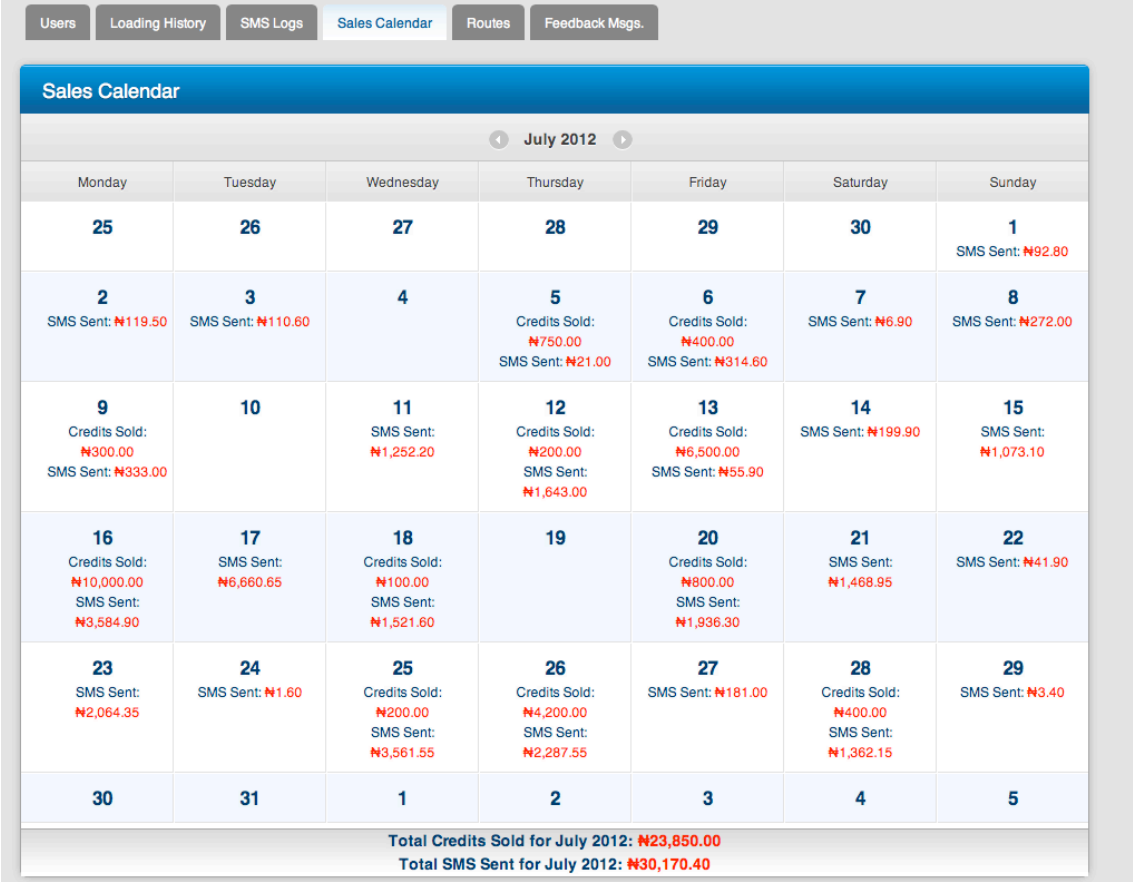


Figure 13. Admin panel showing the daily sales calendar.

4 FACEBOOK APPLICATION INTEGRATION

Just as we mentioned in the software specifications, Social Network integration is an integral part of driving traffic to the application and increasing the user engagement.

In doing this, we have chosen Facebook (www.facebook.com) as our social network. Facebook is a leading social network with over 900 million active users monthly. The network receives an estimated three billion comments per day and three hundred million photos are added every day. Alexa.com (a website traffic ranker, http://www.alexa.com/topsites) ranks facebook.com as the second most visited website in the world, only toppled by the famous search engine Google.

We will be using Facebook's rich platform of API (Application Programming Interface) and plugins to integrate our SMS application.

4.1 The Facebook Platform

The Facebook social platform enables third-party developers to integrate their applications with Facebook. The platform provides a set of API and plugins which developers can plug in to achieve this aim.

The Facebook platform was officially launched in May 2007 and was intended to be a way to drive user engagement on the social site. The platform has been a moving target, always being updated with new APIs, plugins and rules. It has morphed from an early idea of allowing developers build games, apps and adverts to a wider specification that allows developers embed Facebook tags on their site and monitor the user engagement on their site.

The Facebook platform now allows us (third-party developers) to integrate our apps into Facebook or integrate Facebook plugins into our website and even build mobile apps on Facebook mobile.

Most of Facebook's social plugins are implemented with an iFrame (inline frame) code snippets. An iframe is an HTML tag (<iframe>) used to embed another (external/internal) HTML document

within the current HTML document. Below is a sample iframe code snippet used by Facebook. The snippet embeds the Facebook “Like Button” on a website.

```
<html>
    <head>
      <title>My Great Web page</title>
    </head>
    <body>
       <iframe src="https://www.facebook.com/plugins/like.php?href=YOUR_URL"
        scrolling="no" frameborder="0"
        style="border:none; width:450px; height:80px"></iframe>
    </body>
 </html>
```

Figure 14. “Like Button” iFrame tag snippet.

For the rest of the social plugins, Facebook uses what it calls an “XFBML”(extended Facebook Markup Language). For plugins that use the XFBML, Facebook’s JavaScript SDK is required to process the XFBML code. XFBML is a set of XML elements that are included in HTML pages to display the social plugin. As a rule of thumb, the XFBML code provides more flexibility, control and configuration over the iframe codes.
Below is a sample XFBML code snippet that embeds the “Activity Feed” plugin.

```
<fb:activity
site="jerrycain.com"
action="critiqueapp:despise,critiqueapp:review,critiqueapp:grade">
</fb:activity>
```

Figure 15. Example of XFBML code.

4.2 Facebook Social Plugins for Websites

The Facebook platform provides social plugins for integrating Facebook into your website. The social plugins include:

- The “Like Button”.

The Facebook “Like Button” enables your users to share a web page on your site with their Facebook friends with just a single click. The web page they share may be an article you wrote, a product you are selling, a news item you posted or generally any piece of content on your site. The “Like Button” is a form of recommendation a user shares with his network of friends on Facebook. Whatever the user likes, shows up on his Facebook timeline/wall and is also broadcasted to his friends’ activity feed. (Facebook Engineering Team, 2010, Date of retrieval 16.7.2012**).**

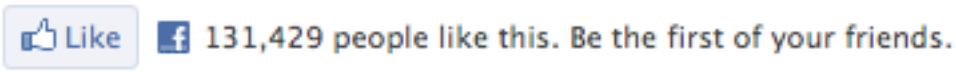


Figure 16: An example of a "Like Button" on an external website. (Facebook Engineering Team, 2010, Date of retrieval 16.7.2012**.)**

- Activity Feed.

The Activity Feed plugin allows you to show visitors of your site a stream of recent "likes" and comments from their friends on your site. Facebook has recently extended this plugin to third party developers to define actions or verbs of their own. Actions like "listened to"(spotify.com uses this verb to show what sound tracks users are listening to), "read", "shared", "attending"(related to an event invitation), "likes", "commented", "recommend" are already defined by default in this plugin. (Facebook Engineering Team, 2010, Date of retrieval 16.7.2012**).**

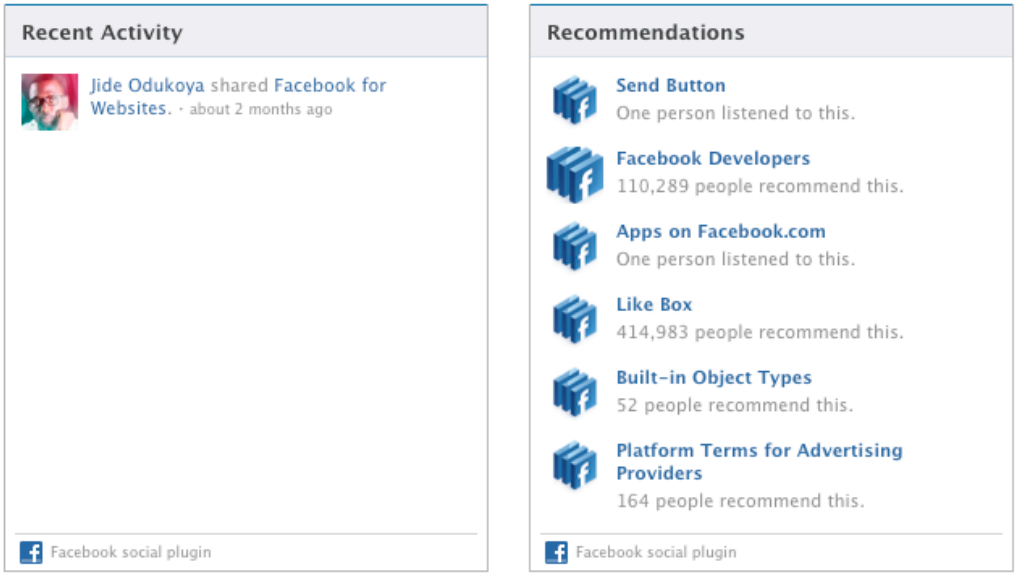


Figure 17: An example of the Activity Feed plugin on an external website. (Facebook Engineering Team, 2010, Date of retrieval 16.7.2012**.)**

- Comments Box

The Comments Box allows third-party developers to add commenting feature on their website. This comes with the ability to moderate comments, mark some as spam and define the comment ordering. Comments made on the external site using this plugin are easily shared with the friends

of the commenter. The comments appear on their friends' news feed, making it easy for the friends to follow up on the discussion, participate by also dropping a comment or just by "liking" it. All the activity generated by the comment plugin is synchronized across Facebook and on the comments box on the external site. (Facebook Engineering Team, 2010, Date of retrieval 16.7.2012**).**

Figure 18. An example of the Comments Box on techcrunch.com. (Burns, M. 2012, Date of Retrieval 23.7.2012.)

- Authentication Plugin

With a user base of over 900 million users, authentication and authorization becomes a big and complex business for Facebook. Facebook now serves as an online directory for searching for long lost friends, colleagues, relations and the Facebook platform provides easy access to this data through the Social Authentication plugin.

The Authentication plugin is composed of two components: the user registration and the sign-in component. With this plugin, third party developers can use the login and registration feature on their external websites. The visitors of the sites where this plugin is used, no longer need to fill in yet another registration form or remember another username and password to be used on the site. They can sign in to the site using their Facebook credentials and have the registration form already prefilled with their personal details from Facebook. Also, if they are already logged in to Facebook and visit the external site, the external site automatically identifies their signed in details, mitigating the need for multiple logins on different websites.

Facebook platform uses OAuth 2.0 protocol for user authentication and authorization. The OAuth is an open protocol that allows a user to grant access to his private resources on one site to another site (Hammer-Lahav, 2007, OAuth Introduction.). OAuth's use case scenario is along the lines of: allowing a user grant access to his pictures on flickr.com to the Kodak.com printing service while still being able to restrict access to his contact details, camera specifications and other private data. OAuth also gives the user the ability to revoke privileges from certain applications.

In order to use the Authentication plugin, third party developers have to register their website with Facebook. When the website is registered Facebook provides the developer with an App ID and App Secret which is used to identify and authenticate requests from the website. Facebook's JavaScript SDK is used to make the requests and responses necessary to use this plugin.

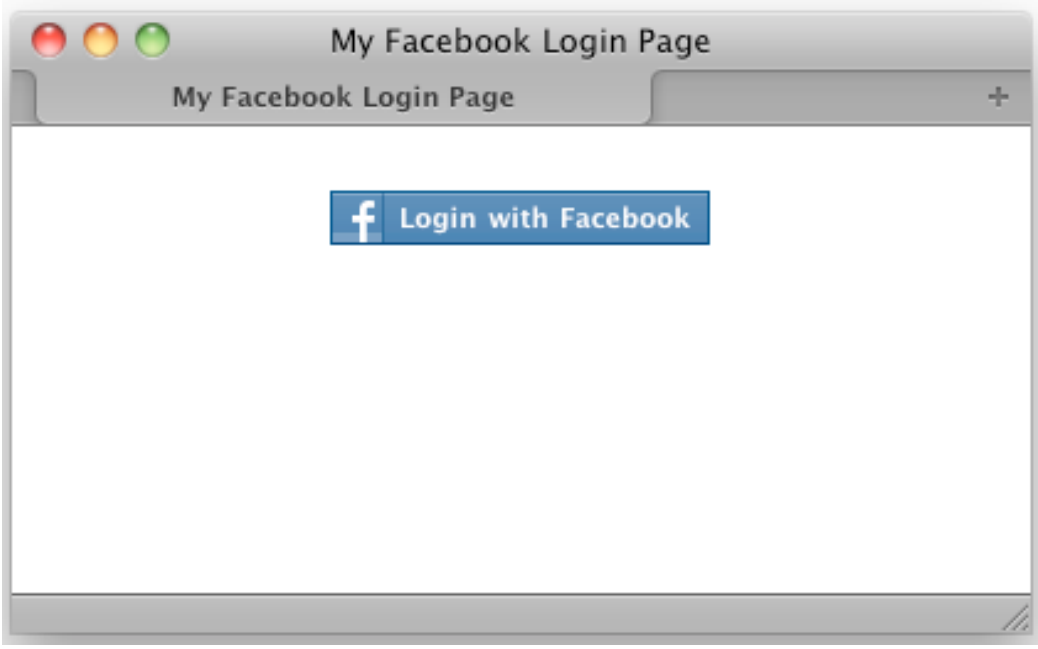


Figure 19. Facebook Login Button on an external website. (Facebook Engineering Team, 2010, Date of retrieval 16.7.2012.)

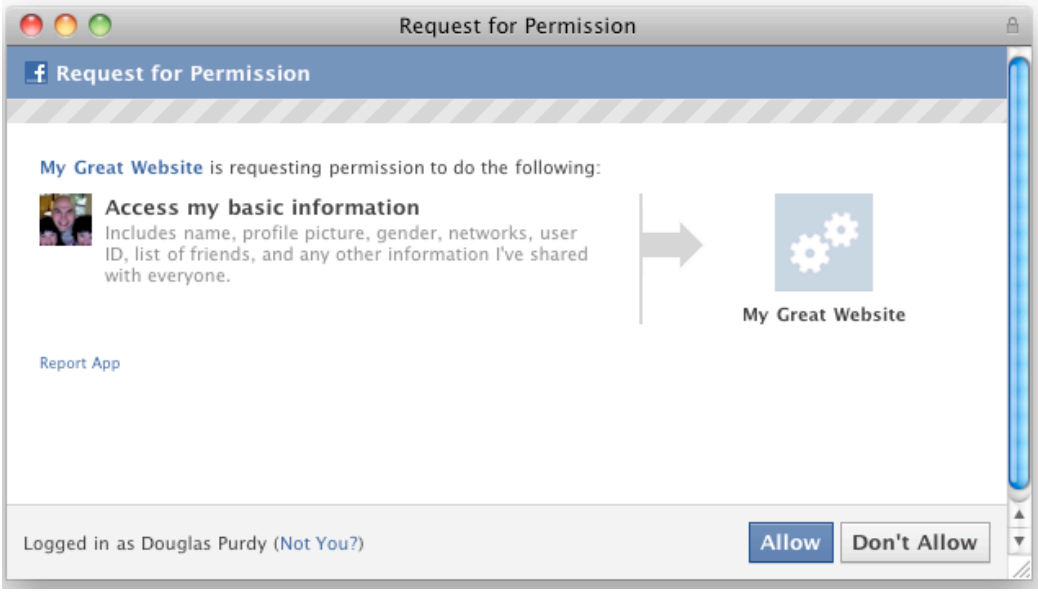


Figure 20. Granting access to a third party website, the OAuth way. (Facebook Engineering Team, 2010, Date of retrieval 16.7.2012.)

4.3 Facebook Apps

On the other side of the coin, integrating a web application or website into Facebook requires creating a Facebook app for it. Facebook Apps are implemented in HTML, so they are fairly straightforward. In Facebook's term, these are called "Canvas Page". A canvas page is an external web application that is loaded in the context of Facebook.

A Canvas page is basically a container iframe used for loading external web apps within the standard Facebook chrome. To be able to use the Canvas page, third party developers need to register their website on Facebook, to obtain an App ID and App Secret used to identify the app. The Canvas page is then provided with a Canvas URL (which is the website to load) that contains the HTML, JavaScript and CSS that make up the app.

4.4 Implementing Facebook Integration

In implementing the integration between Facebook and the SMS app, we took out the following steps:

- Registered the SMS app website on Facebook.

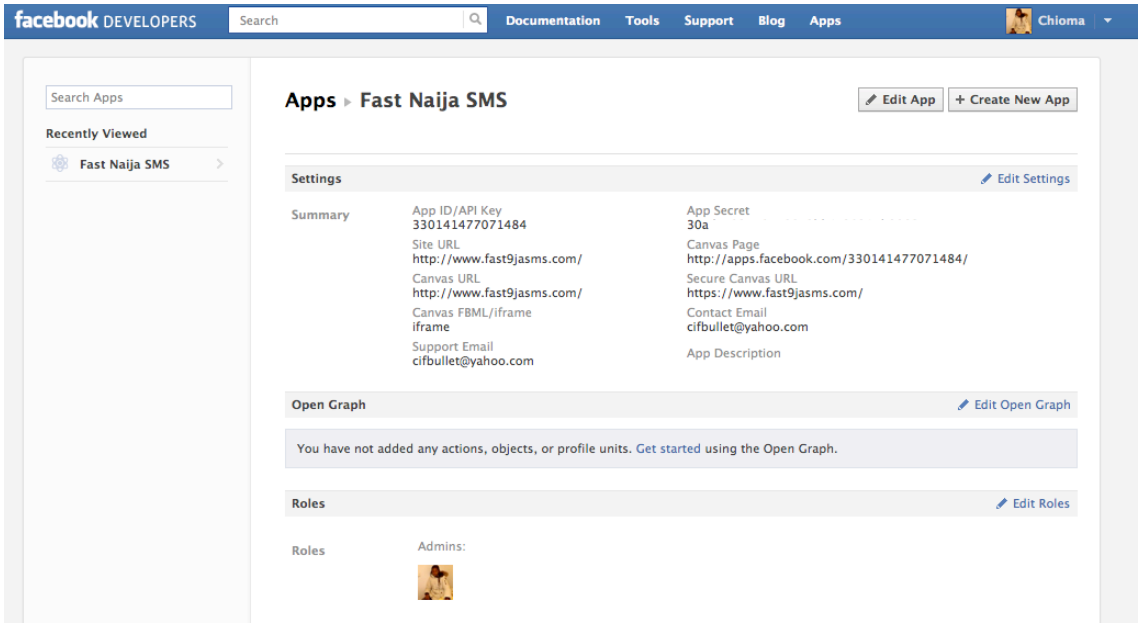


Figure 21. Apps Settings Page. (Facebook, 2012, Date of retrieval 16.7.2012.)

- Created a Facebook page that will host the SMS app.

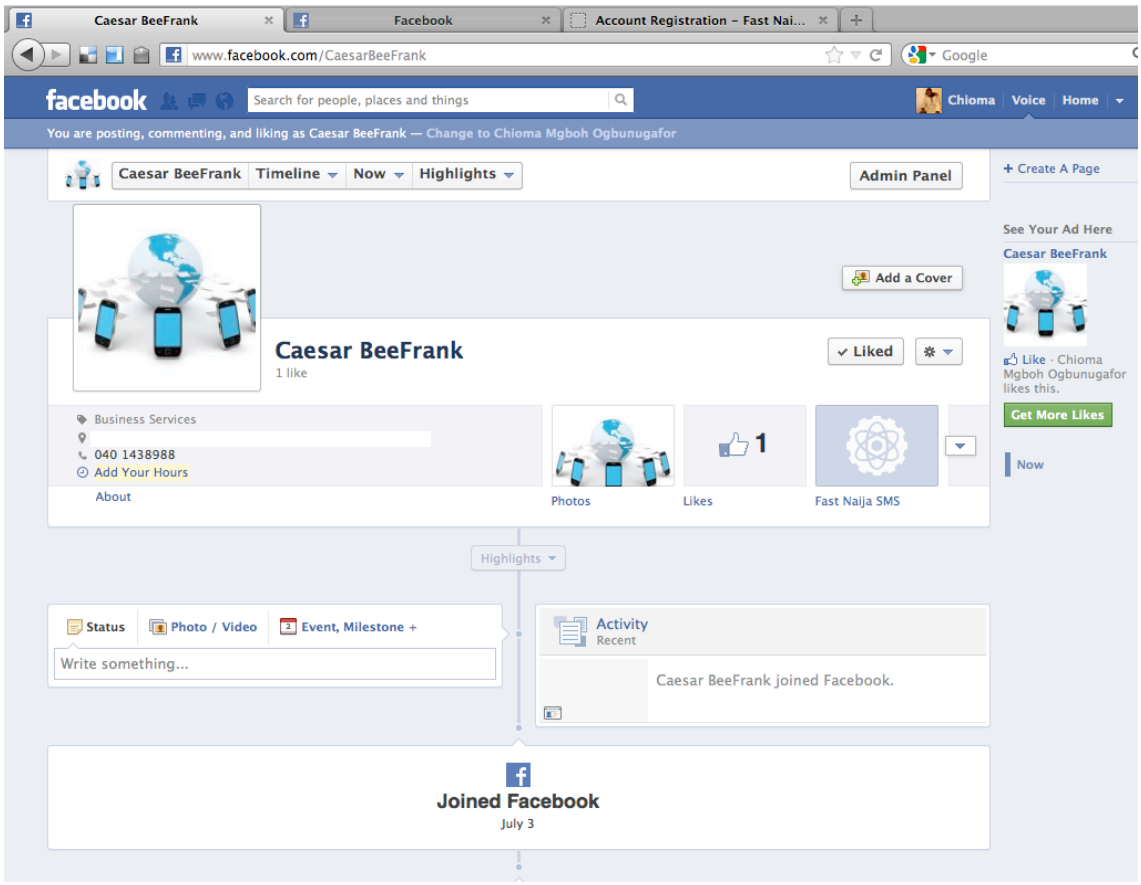


Figure 22. Facebook Page for the SMS App. (Facebook, 2012, Date of retrieval 16.7.2012.)

- Created a Facebook canvas app
- Embedded the canvas app as a tab on our previously created page.

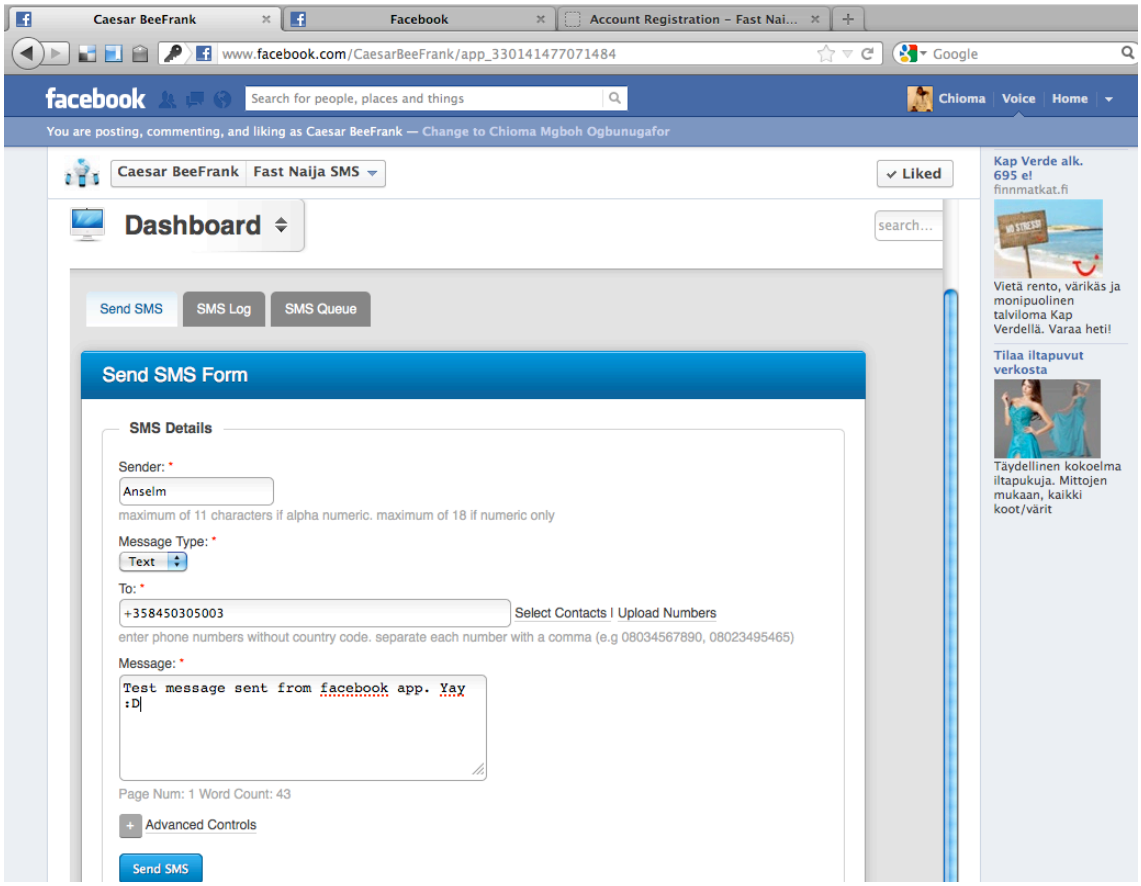


Figure 23. Facebook App for the SMS App. (Facebook, 2012, Date of retrieval 16.7.2012.)

After all this had been done, we now sought to integrate Facebook in the SMS website. We used the Social Authentication plugin to accomplish this. With this, users could register with their Facebook profile and also sign in with it.

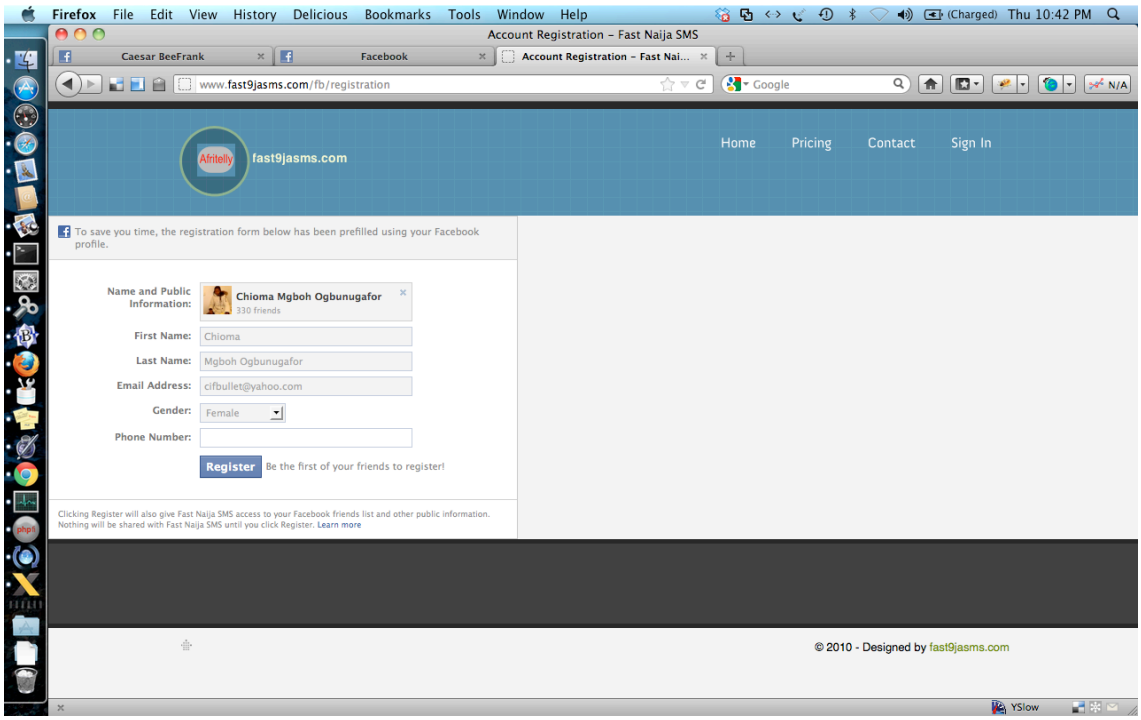


Figure 24. Registration page using the Facebook Social Authentication plugin. (Facebook, 2012, Date of retrieval 16.7.2012.)

5 TESTING AND DOCUMENTATION

Testing is a relevant part of any application development. With testing we can confirm that the requirements of the application are met and passed. Testing provides us with information about the quality of the application, the requirements that are currently met and bugs which were found.

There are different software testing methodologies. We applied 'Regression Testing' while developing the application. Regression testing deals mainly with testing all the components that makes up a system. In regression testing, the system is continually tested while development goes on. Regression testing flags old bugs that have been resolved previously. If these old bugs reappear while new components are added to the system, then we try to fix and resolve them before moving on with new features. Regression testing also flags new bugs creeping in.

5.1 Introducing Selenium

Selenium is a web application testing system that automates browser events and interacts with a web page's elements. It performs click, content input, response verification and so many other events and interactions. It outputs the results of its automation to a log file and also warns of any test that might have failed.

Selenium is also used for testing complex AJAX based web user interfaces. As we will be using AJAX call-backs in many parts of our application, this will be a great addition to our testing toolbox. The diagram below shows a simplified architectural representation of how Selenium works.

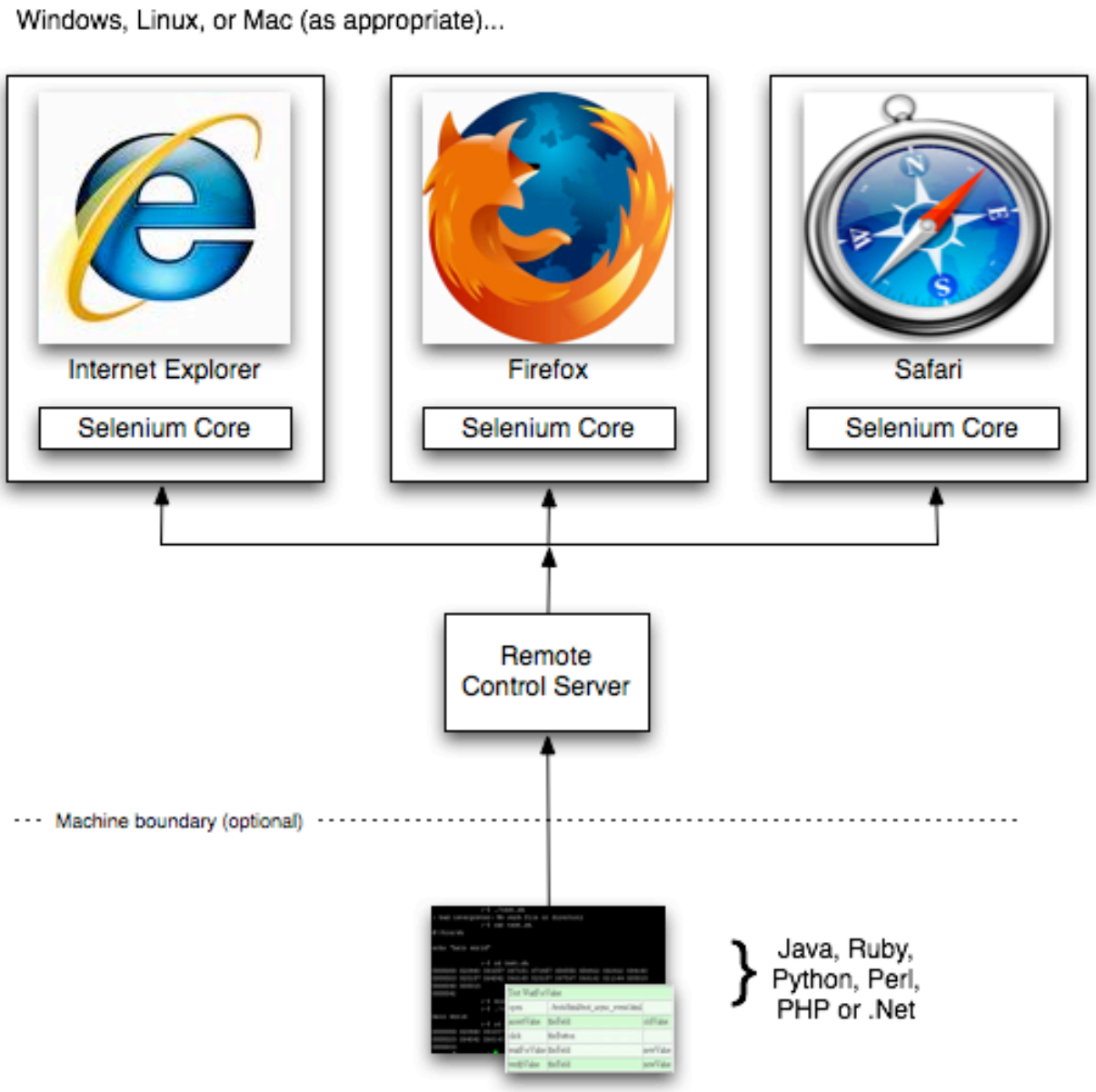
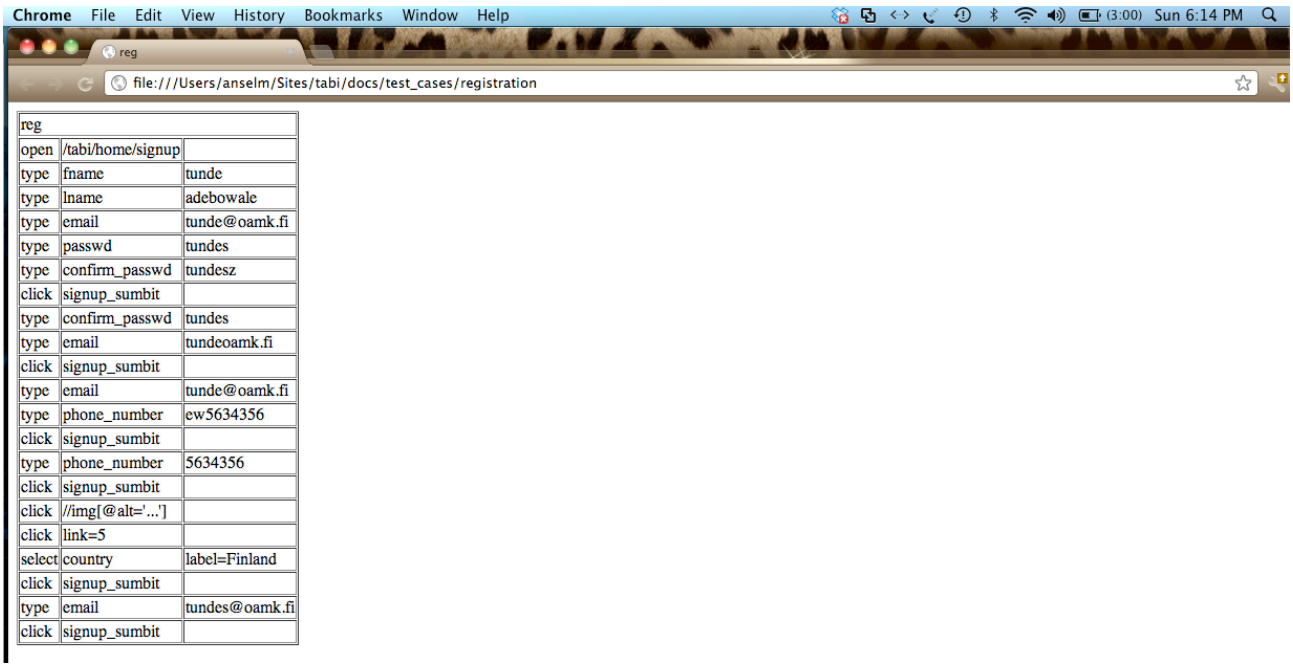


Figure 25. Selenium's Architectural Flow. (Selenium. 2012. Date of retrieval 22.5.2012.)

5.2 Selenium Test Scripts

In order to get Selenium run our automations, we need to use the Selenium Remote Control tool to feed in sample input and expected output. The Selenium Remote Control tool uses an expressive language for creating the test cases you need. The image below shows a sample test case we created for testing the user registration process.



reg		
open	/tabi/home/signup	
type	fname	tunde
type	lname	adebowale
type	email	tunde@oamk.fi
type	passwd	tundes
type	confirm_passwd	tundesz
click	signup_sumbit	
type	confirm_passwd	tundes
type	email	tundeoamk.fi
click	signup_sumbit	
type	email	tunde@oamk.fi
type	phone_number	ew5634356
click	signup_sumbit	
type	phone_number	5634356
click	signup_sumbit	
click	//img[@alt='...']	
click	link=5	
select	country	label=Finland
click	signup_sumbit	
type	email	tundes@oamk.fi
click	signup_sumbit	

Figure 26. Sample test case page.

5.3 Alpha Testing

Before a software is released for public consumption, it is first released to a control group called the alpha testers. This group is a representation of the actual audience for whom the software was designed for. This group is normally small and can range from ten people to hundreds of people.

After the alpha testing has been done, all identified bugs and failure points are worked on and fixed. After this stage of testing, some software producers still go on to have a second round called the beta testing while others go on to release the first version of the software. Your mileage may vary depending on the size of the software users and the implication or risk associated with the software failing.

In our case, we ran an alpha testing period to a group of 15 friends before releasing the first version of our software. Bugs were reported informally to me and I took notes of them to fix them later. No bug reporting tools like Mantis (http://www.mantisbt.org/) or Jira (http://atlassian.com/) were used.

5.4 Software Documentation

In any software project, the documentation is an important aspect of developing software. The documentation of software architecture and code ensures that the software is maintainable; its code is reusable and understood by other programmers who may not have been working on the project from the onset.

There are tools that generate code documentation from source. They output the documentation in different formats like PDF (Portable Document Format), HTML, Microsoft Word Documents and even CHM files (Microsoft Compiled HTML). In this project we used PHPDoc for generating documentation from the source code.

PHPDoc, which is just a port of JavaDoc, inherits the commenting and tag syntax from JavaDoc. PHPDoc uses the DocBlock commenting syntax.

6 CONCLUSIONS

So far we have been able to accomplish the aims we set out for at the beginning of the project. The SMS application was designed and architectured from ground up, the functionalities were coded, tested and deployed to a web server. Users have signed up and are already using the service. It has been a nice learning experience and also a challenging one.

We have also learnt a lot along the way. We learnt about the technological stacks of a web application, the bits and pieces that make up a web application. We learnt about social networks and how they drive up user engagement on a website. We also learnt how to integrate the basic parts of a social network to our web app.

6.1 Possible Future Improvements

One delightful fact about software is that there is always room for improvement. No matter how good a piece of software is, there is always something that can be tweaked to improve the usability, user experience and performance of the software.

When the development on the SMS app began, the HTML5 buzz had not caught on then. Its specification was still at the draft stage and not many browsers had HTML5 support back then (seems like ages ago). But as of the current trend, HTML5 is a must now. All prominent browsers now support HTML5.

For the future, we could refit most of the form elements used with the more user-friendly HTML5 elements. Of particular note is the file upload element, where we had to use an iframe to embed the file upload element so that we do not refresh the page when an upload is done. This hack would not be needed if we used the HTML5 file upload element. This is just one example of how HTML5 can improve the user experience of the app.

In the SMS app we developed, there was no search feature built in. In future versions of the app, this should be a must have feature. We could implement a basic search system or we could take it further by installing the highly scalable full-text search platform ApacheSolr, the Java based front end to the Lucene search engine.

LIST OF SOURCES

Atkinson, J. 2012. The Outernet. Date of retrieval 4.5.2012 http://wronghands1.wordpress.com/2012/03/01/the-outernet/.

Burns, M. 2012. Google Affirms Nexus 7's main fault with this adorable commercial. Date of Retrieval 23.7.2012 http://techcrunch.com/2012/07/23/google-affirms-the-nexus-7s-main-fault-with-this-adorable-commercial/

EllisLab. 2011. CodeIgniter User Guide Version 2.1.2. Date of Retrieval 12.3.2012 http://codeigniter.com/user_guide/overview/at_a_glance.html.

Facebook Engineering Team. 2010. Caesar BeeFrank. Date of retrieval 16.7.2012 http://www.facebook.com/CaesarBeeFrank.

Facebook Engineering Team. 2010. Facebook Developers**.** Internal Source. Date of retrieval 16.7.2012 https://developers.facebook.com/apps/330141477071484.

Facebook Engineering Team. 2010. Facebook for websites**.** Date of retrieval 16.7.2012 https://developers.facebook.com/docs/guides/web/.

Gamma, E., Helm, R., Johnson, R. & Vlissides, J. 1995. Design Patterns: Elements of Reusable Object Oriented Software. Boston, MA: Addison-Wesley. Page 4.

Hammer-Lahav, E. 2007. OAuth Introduction. Date of Retrieval 16.7.2012 http://oauth.net/about/.

Hope, P. & Walther, B. 2009. Web Security Testing Cookbook: Systematic Techniques to Find Problems Fast. Sebastopol, CA: O'Reilly. Page 12.

Sayed, Y. H. & Sayed, I. H. 2006. Deploying .NET Applications: Learning MSBuild and ClickOnce. New York: Apress. Page 3.

Selenium's Architectural Flow. (Selenium. 2012. Selenium Remote-Control. Date of retrieval 22.5.2012 http://seleniumhq.org/projects/remote-control/.)

Soltano, S. 2012 . Web server technology usage reports**.** Date of retrieval 28.4.2012 http://w3techs.com/blog/entry/web_server_technology_usage_reports

Soltano, S. 2012**.** Usage of server-side programming languages for websites. Date of retrieval 28.4.2012 http://w3techs.com/technologies/overview/programming_language/all

Wikipedia. 2005. Social Networking Service. Date of Retrieval 12.1.2012 http://en.wikipedia.org/wiki/Social_networking_service.

APPENDICES

SVN logs follow below

httpdocs# svn log -r 1:head
---
r1 | sloami | 2011-02-11 14:32:14 +0100 (Fri, 11 Feb 2011) | 1 line

importing tabi project
---
r2 | root | 2011-02-11 16:05:39 +0100 (Fri, 11 Feb 2011) | 1 line

sample config files
---
r3 | sloami | 2011-02-11 16:34:11 +0100 (Fri, 11 Feb 2011) | 1 line

home page wordings
---
r4 | sloami | 2011-02-12 08:17:29 +0100 (Sat, 12 Feb 2011) | 1 line

fixed billing for undelivered sms and more account info for the user
---
r5 | root | 2011-02-12 08:21:21 +0100 (Sat, 12 Feb 2011) | 1 line

add htaccess for linux environments
---
r6 | sloami | 2011-02-12 11:17:47 +0100 (Sat, 12 Feb 2011) | 1 line

separate login page done, instead of fancy box implementation
---
r7 | sloami | 2011-02-12 11:32:21 +0100 (Sat, 12 Feb 2011) | 1 line

forgot password functionality started
---
r8 | sloami | 2011-02-12 14:41:23 +0100 (Sat, 12 Feb 2011) | 1 line

password reset link sending done
---
r9 | sloami | 2011-02-12 15:33:52 +0100 (Sat, 12 Feb 2011) | 1 line

passwd reset link validating
---
r10 | sloami | 2011-02-12 15:36:09 +0100 (Sat, 12 Feb 2011) | 1 line

passwd reset link validating
---
r11 | sloami | 2011-02-12 15:40:04 +0100 (Sat, 12 Feb 2011) | 1 line

passwd reset link validating

---
r12 | sloami | 2011-02-12 15:41:16 +0100 (Sat, 12 Feb 2011) | 1 line

passwd reset link validating
---
r13 | sloami | 2011-02-12 15:49:16 +0100 (Sat, 12 Feb 2011) | 1 line

passwd reset link validating- debugging
---
r14 | sloami | 2011-02-12 15:51:47 +0100 (Sat, 12 Feb 2011) | 1 line

passwd reset link validating- debugging
---
r15 | sloami | 2011-02-12 15:54:14 +0100 (Sat, 12 Feb 2011) | 1 line

passwd reset link validating- debugging
---
r16 | sloami | 2011-02-12 16:13:08 +0100 (Sat, 12 Feb 2011) | 1 line

passwd reseting done
---
r17 | sloami | 2011-02-12 16:14:44 +0100 (Sat, 12 Feb 2011) | 1 line

passwd reseting done
---
r18 | sloami | 2011-02-12 16:18:17 +0100 (Sat, 12 Feb 2011) | 1 line

passwd reseting done - fix to reset button event attaching
---
r19 | sloami | 2011-02-12 17:20:57 +0100 (Sat, 12 Feb 2011) | 1 line

passwd reseting done - fix to reset button event attaching
---
r20 | sloami | 2011-02-13 12:10:21 +0100 (Sun, 13 Feb 2011) | 1 line

fix to fancybox login on sign up page
---
r21 | sloami | 2011-02-14 08:12:21 +0100 (Mon, 14 Feb 2011) | 1 line

payment wording change
---
r22 | sloami | 2011-02-15 06:53:01 +0100 (Tue, 15 Feb 2011) | 1 line

fix to sign up bug. db error, column count doesn't match
---
r23 | sloami | 2011-02-15 06:56:18 +0100 (Tue, 15 Feb 2011) | 1 line

fix to sms sending smarty template
---
r24 | sloami | 2011-02-15 11:36:31 +0100 (Tue, 15 Feb 2011) | 1 line

google analytics installed
---
r25 | sloami | 2011-02-15 11:54:40 +0100 (Tue, 15 Feb 2011) | 1 line

google analytics installed to sign up page
---
r26 | sloami | 2011-02-17 08:58:25 +0100 (Thu, 17 Feb 2011) | 1 line

minor changes: copy text change for expired session, img sample for csv uploads
---
r27 | sloami | 2011-02-17 09:36:02 +0100 (Thu, 17 Feb 2011) | 1 line

added csv sample files, changed copy text on Payment details
---
r28 | sloami | 2011-02-18 18:03:28 +0100 (Fri, 18 Feb 2011) | 1 line

sender id restriction to 11 chars and saving to sms_log table
---
r29 | sloami | 2011-02-19 14:05:24 +0100 (Sat, 19 Feb 2011) | 1 line

message type & sender ID done
---
r30 | sloami | 2011-02-19 14:41:22 +0100 (Sat, 19 Feb 2011) | 1 line

message type debugging
---
r31 | sloami | 2011-02-19 14:48:54 +0100 (Sat, 19 Feb 2011) | 1 line

message type debugging
---
r32 | root | 2011-02-23 13:12:14 +0100 (Wed, 23 Feb 2011) | 1 line

fixed wrong pagination URL for users view
---
r33 | root | 2011-02-26 13:26:22 +0100 (Sat, 26 Feb 2011) | 1 line

fix to js bug on group selection on contact page
---
r34 | sloami | 2011-04-07 16:29:25 +0200 (Thu, 07 Apr 2011) | 1 line

copy text changes to home pages
---
r35 | sloami | 2011-05-30 19:59:30 +0200 (Mon, 30 May 2011) | 1 line

theuniquesms.com copytext changes
---
r36 | sloami | 2011-05-30 20:05:18 +0200 (Mon, 30 May 2011) | 1 line

theuniquesms.com copytext changes

---
r37 | sloami | 2011-05-30 20:07:13 +0200 (Mon, 30 May 2011) | 1 line

theuniquesms.com copytext changes
---
r38 | sloami | 2011-05-30 20:17:50 +0200 (Mon, 30 May 2011) | 1 line

theuniquesms.com copytext changes
---
r39 | sloami | 2011-05-30 21:48:17 +0200 (Mon, 30 May 2011) | 1 line

theuniquesms.com copytext changes
---
r40 | sloami | 2011-06-01 16:23:11 +0200 (Wed, 01 Jun 2011) | 1 line

theuniquesms.com copytext changes
---
r41 | sloami | 2011-06-01 16:30:15 +0200 (Wed, 01 Jun 2011) | 1 line

theuniquesms.com copytext changes
---
r42 | sloami | 2011-06-02 22:50:51 +0200 (Thu, 02 Jun 2011) | 1 line

contacts accordion bug fix
---
r43 | sloami | 2011-06-02 22:49:33 +0200 (Thu, 02 Jun 2011) | 1 line

contacts accordion bug fix
---
r44 | sloami | 2011-06-19 22:09:01 +0200 (Sun, 19 Jun 2011) | 1 line

afritelly img
---
r45 | sloami | 2011-06-22 18:55:58 +0200 (Wed, 22 Jun 2011) | 1 line

credit loading miscalculation fixed
---
r46 | sloami | 2011-06-24 09:16:05 +0200 (Fri, 24 Jun 2011) | 1 line

fixing sms send bug
---
r47 | sloami | 2011-06-24 09:24:31 +0200 (Fri, 24 Jun 2011) | 1 line

added missing files
---
r48 | sloami | 2011-07-14 20:04:29 +0200 (Thu, 14 Jul 2011) | 1 line

sms routes implementation done. template samples created
---
r49 | sloami | 2011-07-14 20:26:45 +0200 (Thu, 14 Jul 2011) | 1 line

fix to sms routes

---

r50 | sloami | 2011-07-14 21:24:30 +0200 (Thu, 14 Jul 2011) | 1 line

adding report controller

---

r51 | sloami | 2011-07-14 22:10:08 +0200 (Thu, 14 Jul 2011) | 1 line

bug fix

---

r52 | sloami | 2011-07-14 22:18:58 +0200 (Thu, 14 Jul 2011) | 1 line

bug fix

---

r53 | sloami | 2011-07-14 22:20:32 +0200 (Thu, 14 Jul 2011) | 1 line

bug fix

---

r54 | sloami | 2011-07-14 22:24:29 +0200 (Thu, 14 Jul 2011) | 1 line

bug fix

---

r55 | sloami | 2011-07-14 22:27:17 +0200 (Thu, 14 Jul 2011) | 1 line

bug fix

---

r56 | sloami | 2011-07-15 23:31:16 +0200 (Fri, 15 Jul 2011) | 1 line

bug fix to groups and contacts not showing

---

r57 | sloami | 2011-07-15 23:36:06 +0200 (Fri, 15 Jul 2011) | 1 line

css class fix

---

r58 | sloami | 2011-07-16 14:23:38 +0200 (Sat, 16 Jul 2011) | 1 line

admin loading history

---

r59 | sloami | 2011-07-16 14:26:18 +0200 (Sat, 16 Jul 2011) | 1 line

added missing templates

---

r60 | sloami | 2011-07-16 14:33:55 +0200 (Sat, 16 Jul 2011) | 1 line

admin loading history

---

r61 | sloami | 2011-07-17 12:14:00 +0200 (Sun, 17 Jul 2011) | 1 line

sms logs and transactions

---
r62 | sloami | 2011-07-17 12:22:10 +0200 (Sun, 17 Jul 2011) | 1 line

sms logs and transactions
---
r63 | sloami | 2011-07-17 12:31:53 +0200 (Sun, 17 Jul 2011) | 1 line

sms logs and transactions
---
r64 | sloami | 2011-07-22 21:50:43 +0200 (Fri, 22 Jul 2011) | 1 line

sales calendar feature
---
r65 | sloami | 2011-07-22 21:53:00 +0200 (Fri, 22 Jul 2011) | 1 line

sales calendar feature
---
r66 | sloami | 2011-07-23 08:56:36 +0200 (Sat, 23 Jul 2011) | 1 line

sales calendar feature
---
r67 | sloami | 2011-07-23 09:00:03 +0200 (Sat, 23 Jul 2011) | 1 line

sales calendar feature
---
r68 | sloami | 2011-07-25 17:26:45 +0200 (Mon, 25 Jul 2011) | 1 line

sales calendar feature
---
r69 | sloami | 2011-07-27 18:08:37 +0200 (Wed, 27 Jul 2011) | 1 line

bug fix to loading history
---
r70 | sloami | 2011-07-27 18:11:40 +0200 (Wed, 27 Jul 2011) | 1 line

bug fix to loading history
---
r71 | sloami | 2011-07-27 18:47:05 +0200 (Wed, 27 Jul 2011) | 1 line

bug fix to credit loading undercharging users
---
r72 | sloami | 2011-07-28 20:27:15 +0200 (Thu, 28 Jul 2011) | 1 line

contact editing
---
r73 | sloami | 2011-08-03 16:05:11 +0200 (Wed, 03 Aug 2011) | 1 line

contacts birthday field
---
r74 | sloami | 2011-08-28 11:21:27 +0200 (Sun, 28 Aug 2011) | 1 line

contacts import now supports birthday & misc fixes here and there
---
r75 | sloami | 2012-07-10 19:46:21 +0200 (Tue, 10 Jul 2012) | 1 line

facebook registration/login button added
---
r76 | sloami | 2012-07-10 20:41:15 +0200 (Tue, 10 Jul 2012) | 1 line

facebook registration form added
---
r77 | sloami | 2012-07-10 20:42:54 +0200 (Tue, 10 Jul 2012) | 1 line

facebook social plugins controller
---
r78 | sloami | 2012-07-10 20:47:09 +0200 (Tue, 10 Jul 2012) | 1 line

facebook registration form added, app ID fix
---
r79 | sloami | 2012-07-10 20:51:19 +0200 (Tue, 10 Jul 2012) | 1 line

facebook registration redirect-uri fix
---
r80 | sloami | 2012-07-10 20:53:25 +0200 (Tue, 10 Jul 2012) | 1 line

facebook registration redirect-uri fix
---
r81 | sloami | 2012-07-10 21:17:30 +0200 (Tue, 10 Jul 2012) | 1 line

facebook registration redirect-uri fix
---
r82 | sloami | 2012-07-11 00:11:14 +0200 (Wed, 11 Jul 2012) | 1 line

facebook registration redirect-uri fix
---
r83 | sloami | 2012-07-11 00:12:52 +0200 (Wed, 11 Jul 2012) | 1 line

facebook registration redirect-uri fix
---
r84 | sloami | 2012-07-11 00:14:43 +0200 (Wed, 11 Jul 2012) | 1 line

facebook registration redirect-uri fix
---
r85 | sloami | 2012-07-14 22:33:45 +0200 (Sat, 14 Jul 2012) | 1 line

registration request data capture
---
r86 | sloami | 2012-07-21 22:02:59 +0200 (Sat, 21 Jul 2012) | 1 line

registration request data capture
