“WIKIPEDIA USABILITY ENHANCEMENTS”

A PROJECT REPORT

Submitted by

Mr. Akshay Gopiramandas Agarwal

in partial fulfilment for the award of the Final Yr. B.Tech Project Credits

of

Final Year B. Tech (CSE) – VIIIth Semester

IN

Computer Science and Engineering

At

उत्कृष्ट तंत्रज्ञानार्थम जन्शक्तये शिक्षणम्

SHRI GURU GOBIND SINGHJI
INSTITUTE OF ENGINEERING AND TECHNOLOGY,
VISHNUPURI, NANDED
(MAHARASHTRA STATE)
PIN 431 606 INDIA

May 2012

“WIKIPEDIA USABILITY ENHANCEMENTS”

A PROJECT REPORT

Submitted by

Mr. Akshay Gopiramandas Agarwal

in partial fulfilment for the award of the Final Yr. B.Tech Project Credits

of

Final Year B. Tech (CSE) – VIIIth Semester

IN

Computer Science and Engineering

At

उत्कृष्ट तंत्रज्ञानार्थम जन्शक्तये शिक्षणम्

SHRI GURU GOBIND SINGHJI
INSTITUTE OF ENGINEERING AND TECHNOLOGY,
VISHNUPURI, NANDED
(MAHARASHTRA STATE)
PIN 431 606 INDIA

May 2012

ABSTRACT

Wikipedia is world's 6th most visited website. It is offered in 289 languages, has a database of more than 16 TB & is used by millions of users across the world. MediaWiki is an open source web application that powers Wikipedia & is developed by the Wikimedia Foundation

For most web properties, a user is considered "converted" when they create a user account. However, Wikipedia does not consider a user "converted" until they have graduated into becoming a "New Wikipedian" by doing at least ten edits into the article namespace.

Currently, there exists no practical way to determine what login or account creation processes have an impact on user conversion. This is a rather severe vulnerability in our capability to determine the best path for editor retention and participation.

This document describes a proposed system whereby much of this data may be gathered going forward as well as improving the overall user experience for logging in and creating accounts. It aims to refactor the existing codebase by making it more modular & extensible. It also demonstrates the great impact that this system has by listing the results of extensive testing. The project follows usability guidelines developed over several years of research work conducted by eminent researchers at various universities.

TABLE OF CONTENTS

TABLE OF CONTENTS

List of Figures

CERTIFICATE

This is to certify that project entitled ***"Wikipedia Usability Enhancements"*** which is submitted by **Akshay Gopiramandas Agarwal (2008BCS154)** in partial fulfillment of the requirement for the award of degree Bachelor of Technology in Computer Science and Engineering to Shri Guru Gobind Singhji Institute of Engineering & Technology, Vishnupuri, Nanded (M. S.) for **Project – II** during academic year **2011-12** and has been carried out under our supervision.

Head
Prof. Dr. U. V. Kulkarni

Guide
Ms. P.D Podgantwar
Assistant Professor

Project Co-Ordinator
(For Project–II in 2011-12)
Mr. A. B.Mandalkar

CSE Department, 15 May 2012

ACKNOWLEDGEMENT

I would like to express my gratitude to all those who helped me to complete this work. I want to thank my guide **Ms. P.D Podgantwar** for her continuous help and generous assistance. She helped in a broad range of issues from giving me direction, helping to find the solutions, outlining the requirements and always having the time to see me.

I am highly grateful to **Mr. Brandon Harris**, Lead Designer, Wikimedia Foundation for mentoring me throughout this project. I am also thankful to **Ms. Sumana Harihareshwara** for managing various aspects of the project. I also wish to thank the entire MediaWiki community for their constant support & encouragement.

I have furthermore to thank **Dr. U.V. Kulkarni**, Head of the Department of Computer Science & Engineering, to encourage me to go ahead and for continuous guidance. I also want to thank **Mr. A.B Mandalkar**, Project Coordinator, for all his assistance and guidance for preparing report.

I would like to thank my colleagues who helped me time to time from preparing report and giving good suggestions. I also extend sincere thanks to all the staff members of Department of Computer Science & Engineering for helping me in various aspects.

Last but not least I am grateful to my parents for all their support and encouragement.

Akshay Agarwal
(2008BCS154)

Chapter 1

Introduction

1.1 Introduction

Wikipedia is world's 6th most visited website. It is offered in 289 languages, has a database of more than 16 TB & is used by millions of users across the world. MediaWiki is an open source web application that powers Wikipedia & is developed by the Wikimedia Foundation. For most web properties, a user is considered "converted" when they create a user account. However, Wikipedia does not consider a user "converted" until they have graduated into becoming a "New Wikipedian" (ten edits into the article namespace). The user interface that handles account creation system involves server side form validation which causes a lot of difficulties for signing up for a new account such as if we choose an existing username then we are not informed about it unless we have submitted the form & then we have to enter all the details again.

Currently, there exists no practical way to determine what login or account creation processes have an impact on user conversion. This is a rather severe vulnerability in our capability to determine the best path for editor retention and participation. There is no system which keeps track of what events are the most fruitful for account creation & which articles have the most impact on encouraging users for creating account. Due to this, we are unable to determine the exit processes for a user after account creation.

1.2 Necessity

Currently, the UI & backend logic are quite mixed up in LoginForm. This structure makes it very difficult for developers to extend it. Also, it does not provide sufficient hooks which can be used in plugins in order to modify the processes. The user interface is now obsolete and does not possess an attractive visual appeal. With the current trends in Web, aesthetics of a site can have significant impacts on its outreach & usability. The current layout of the login page & signup page does not encourage the user to signup with Wikipedia It gives an appearance that the creating a new account is a very time consuming process & hence it turns away the new users. The current account creation flow & the necessary steps are summarized as follows

Current account creation flow

Here is the current flow on the English Wikipedia:

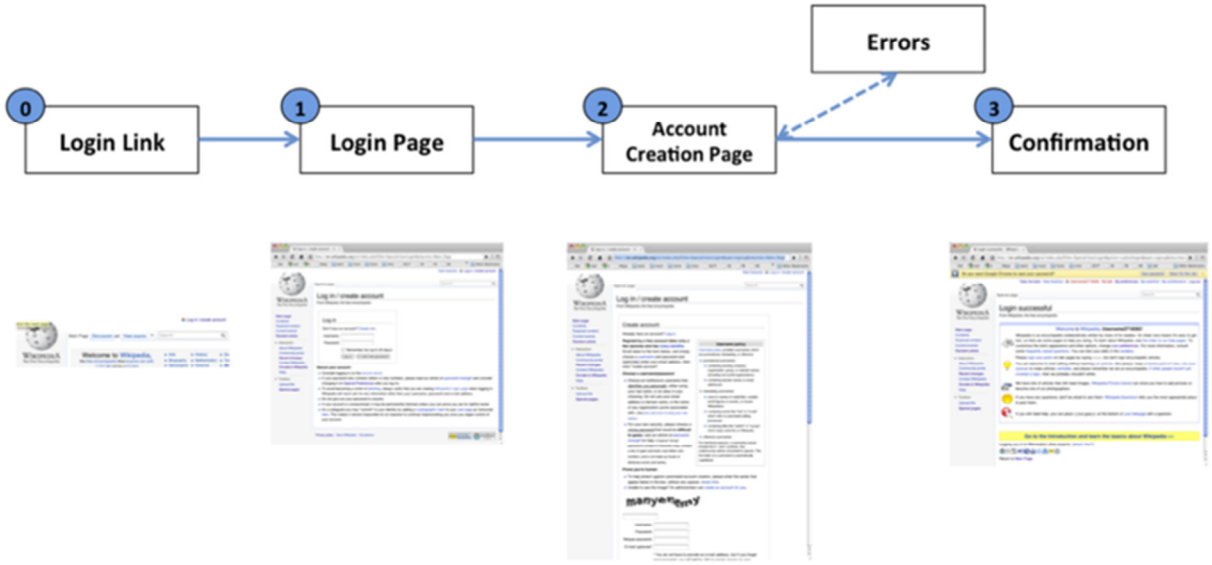

Fig. 1.1 Current User Account Creation Flow

Current steps for account creation:

- 0: User clicks on the Log in/ create account link in the navigation
- 1: User is taken to the Login page, which asks the user to either log in or to create an account
- 2: User is taken to the Account Creation pages, which asks the user to create an account by filling in a form
- 3: User is taken to a Confirmation page

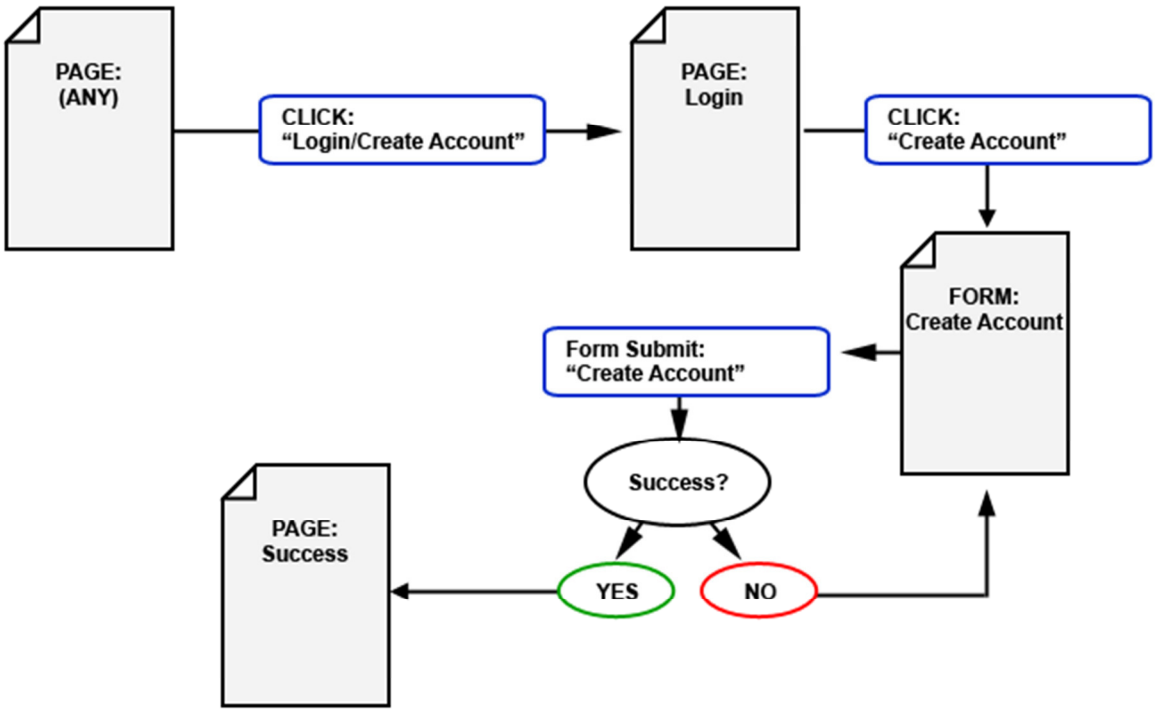

Fig. 1.2 Current steps for Account Creation

1.3 OBJECTIVES

The main purpose of this project is to ensure that the process of account creation (or login) does not wildly interrupt the user's task flow. This will help Wikipedia (& other Wikis) in offering a better user experience & thus increase the number of 'converted' users. It consists of the following objectives

- I propose to take out the entire logic for Login/Account Creation from Special:User Login & put it inside its own class. So now, Special:UserLogin will become a consumer of this Login/Account Creation class & will contain only the presentation logic by implementing the HTMLForm Class.
- APILogin.php & APILogout.php will become consumers of the Login/Account Creation Class.
- The "Source Avenue" for user account creation will be tracked which will help in determining how, when, and why people create accounts. This "source avenue" will be informed to the Login/Account Creation API which will suggest the suitable "Exit Acitivities" for the user.
- The particular "Exit Activity" chosen by the user will be recorded helping us to see which exit activities are better at encouraging conversion.
- The usability of the forms will be enhanced using AJAX by implementing in-situ javascript dialogs, client-side form field validations & tweaking the interaction design.

Chapter 2

Literature Survey

2.1 Website Usability

2.1.1 Augmenting Human Capabilities

Software's real goal should not be to simply process transactions in a system where users are nothing more than data operators who click required buttons to make things happen. Rather, software should work to augment human capabilities, helping us to overcome weaknesses and emphasize our strengths.

A vital way for computers to help users is to direct their attention to a smaller number of important issues instead of overwhelming them with all possible options. For example, BondWorks originally presented 85 attributes of tradable bonds on the search results screen. However, search log analysis showed that nearly three quarters of users were searching on only 10 of the attributes, so the screen was redesigned to focus on those attributes. [1]

2.1.2 Linear Workflow and Wizards

We want to empower users to be creative and accomplish advanced things with our software. But we should also recognize that users sometimes just want to get their tasks done without having to explore numerous options and new ideas.

To speed users through infrequent or complicated tasks, it's often good to present a linear workflow with minimal disruptions or alternatives. Yes, the lack of flexibility can feel constraining, but it can be faster to just power through all the steps instead of having to ponder which steps are needed. Also, the cost of too much freedom is that users have to decide the order of the steps — something that they're often happy enough to delegate to the computer.

Wizards are the classic approach to linear workflows, and several winning applications include nicely designed wizards. The Climate Action Planning Tool uses the simpler approach of offering numbered steps to guide users. An early design included a Before You Start step, but user studies showed that people ignored it. In the final design, the Before activities appear under Step 1: Gather Baseline Energy Consumption Data. Users just want to get going, so you need to be somewhat heavy-handed if setup work is required before starting the real activity.

Wizards don't work for everything. In user testing, Eventbrite developers discovered problems with a wizard for creating new events in the system. Because users' mental models of setting up a new event didn't include all the necessary steps, they often dropped off and didn't complete the wizard. So, the team decided instead to go with a 2-step process in which users first entered all the information and then customized their pages. [1]

2.2 Usability Guidelines

2.2.1 Forget the "Three-Click Rule"

The idea that users will get frustrated if they have to click more than three times to find a piece of content on your website has been around for ages. In 2001, Jeffrey Zeldman, a recognized authority in the web design industry, wrote that the three-click rule "can help you create sites with intuitive, logical hierarchical structures" in his book, *Taking Your Talent to the Web*.

Logically, it makes sense. Of course, users will be frustrated if they spend a lot of time clicking around to find what they need.

But why the arbitrary three-click limit? Is there any indication that web users will suddenly give up if it takes them three clicks to get to what the want?

In fact, most users **won't** give up just because they've hit some magical number. The number of clicks they have to make isn't related to user frustration.

A study conducted by Joshua Porter published on User Interface Engineering found out that users aren't more likely to resign to failure after three clicks versus a higher number such as 12 clicks. "Hardly anybody gave up after three clicks," Porter said. [1]

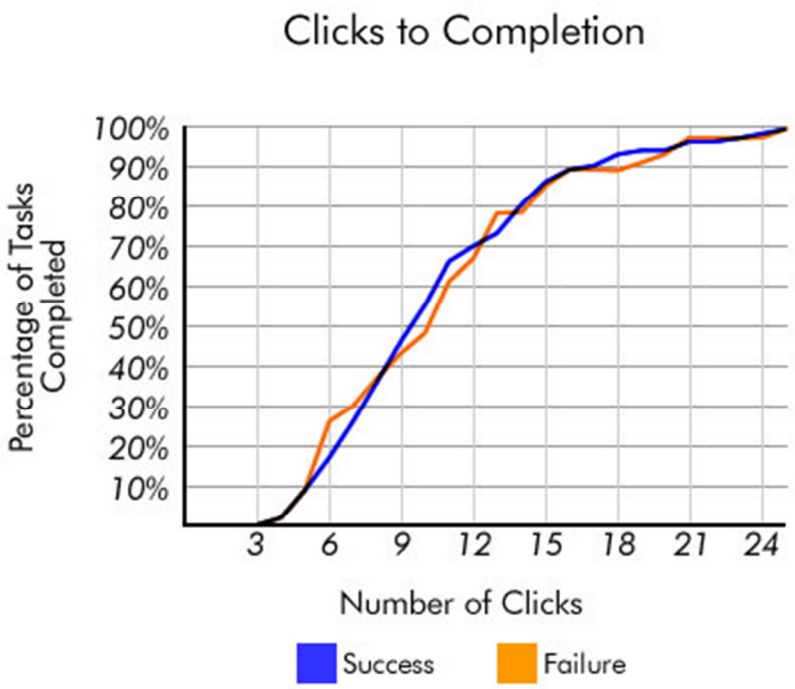

Fig. 2.1 Task Completion vs. Number of Clicks [1]

2.2.2 Enable Content Skimming By Using an F-Shaped Pattern

Dr. Jakob Nielsen, a pioneer in the field of usability, conducted an eye tracking study on the reading habits of web users comprising of over 230 participants. What the research study displayed was that participants exhibited an F-shaped pattern when scanning web content. [4]

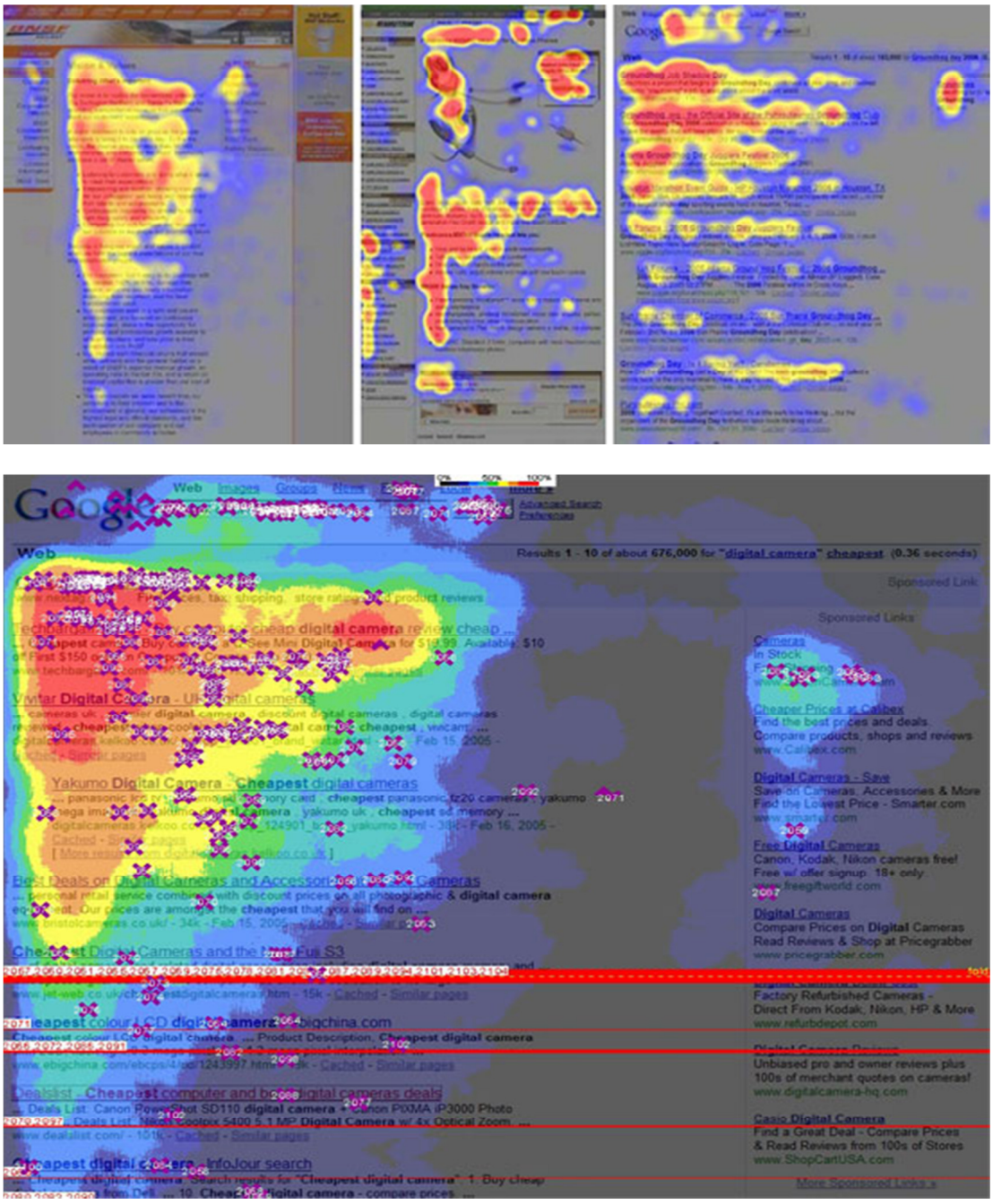

Fig 2.2 F-Shaped Pattern [4]

2.2.3 Don't Make Users Wait: Speed Up Your Website

We're always told that our users are impatient: they hate waiting. Well, that's logical — who likes waiting on purpose? But is there any proof outside of anecdotal evidence that people actually don't like waiting and that page performance affects website users?

Bing, Microsoft's search engine, conducted an analysis to see if there are any correlations between page speed versus numerous performance indicators such as satisfaction, revenue generated per user, and clicking speed. The report showed that a less than 2-second increase of delays in page responsiveness reduced user satisfaction by -3.8%, lost revenue per user of -4.3% and a reduced clicks by -4.3%, among other findings. For a company as large as Microsoft, even a 4.3% drop in revenue can equate to multi-million-dollar losses in profit. [2]

	Distinct Queries/User	Query Refinement	Revenue/User	Any Clicks	Satisfaction	Time to Click (increase in ms)
50ms	-	-	-	-	-	-
200ms	-	-	-	-0.3%	-0.4%	500
500ms	-	-0.6%	-1.2%	-1.0%	-0.9%	1200
1000ms	-0.7%	-0.9%	-2.8%	-1.9%	-1.6%	1900
2000ms	-1.8%	-2.1%	-4.3%	-4.4%	-3.8%	3100

Fig. 2.3 Effect of page load times on user satisfaction [2]

So users, in fact, are impatient: They're less satisfied and will reduce their number of clicks if they wait too long. And if you care about search engine ranking, then the incentive to improve page response times is even greater since Google now factors page speed in their search ranking.

What can you do to improve page performance? Use tools that will help you find performance bottlenecks, use CSS sprites to improve page speed, and utilize benchmarking tools like YSlow to quickly see where you can make quick front-end optimizations.

2.2.4 Make Your Content Easily Readable

Internet users don't really read content online, at least according to a study by Dr. Nielsen on reading behaviors of people on his website. His analysis shows that people only read 28% of the text on a web page and decreased the more text there is on the page. [3]

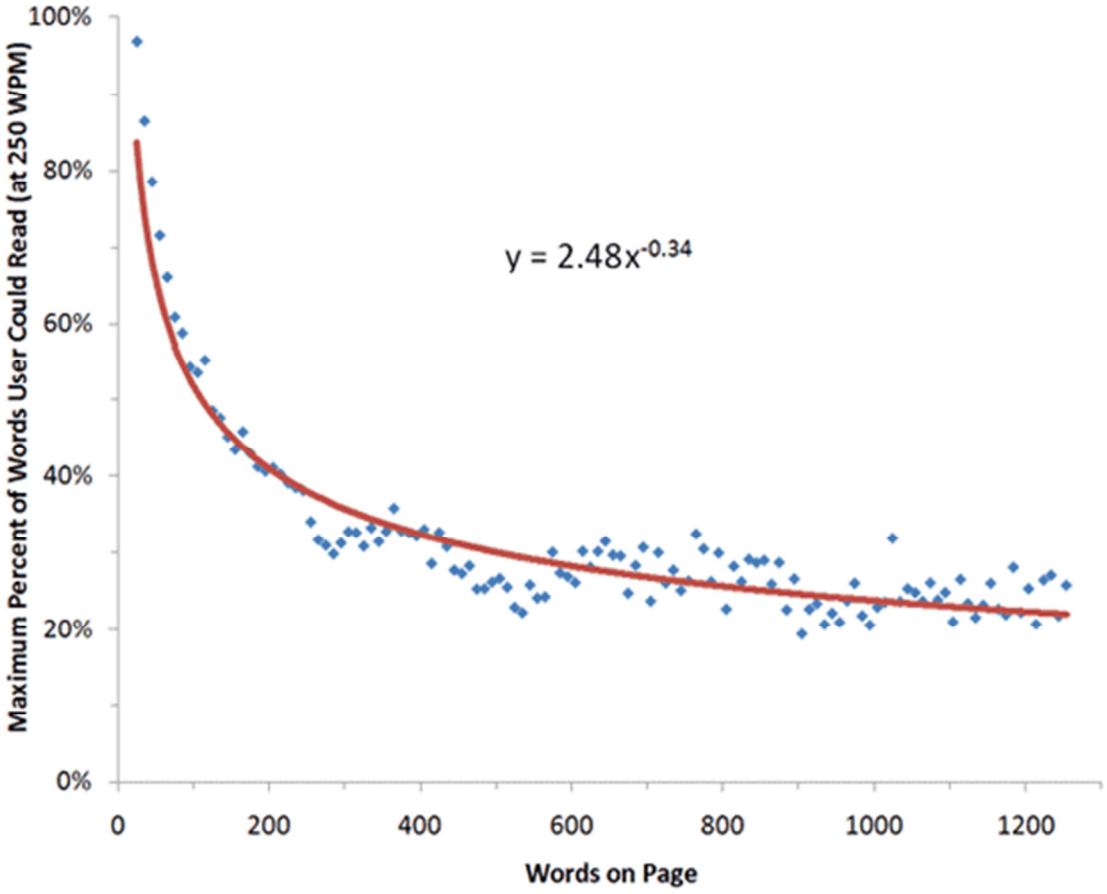

Fig. 2.4 Common reading behaviors [3]

To increase the likelihood of your readers getting the most out of your content, utilize techniques for making content easier to read. Highlight keywords, use headings, write short paragraphs, and utilize lists.

2.2.5 Don’t Worry About "The Fold" and Vertical Scrolling

There has long been a myth that all of your important content should be above "the fold," a term borrowed from newspapers that refers to the area of a web page that can be seen without having to scroll down — first proposed by Jakob Nielsen.

So, are long pages bad? Should we cram everything at the top of our web layouts because people won’t ever read anything below this fold?

The answer is "No" according to a report by Clicktale, a web analytics company. Their results showed that the length of the page has no influence in the likelihood that a user will scroll down the page. [3]

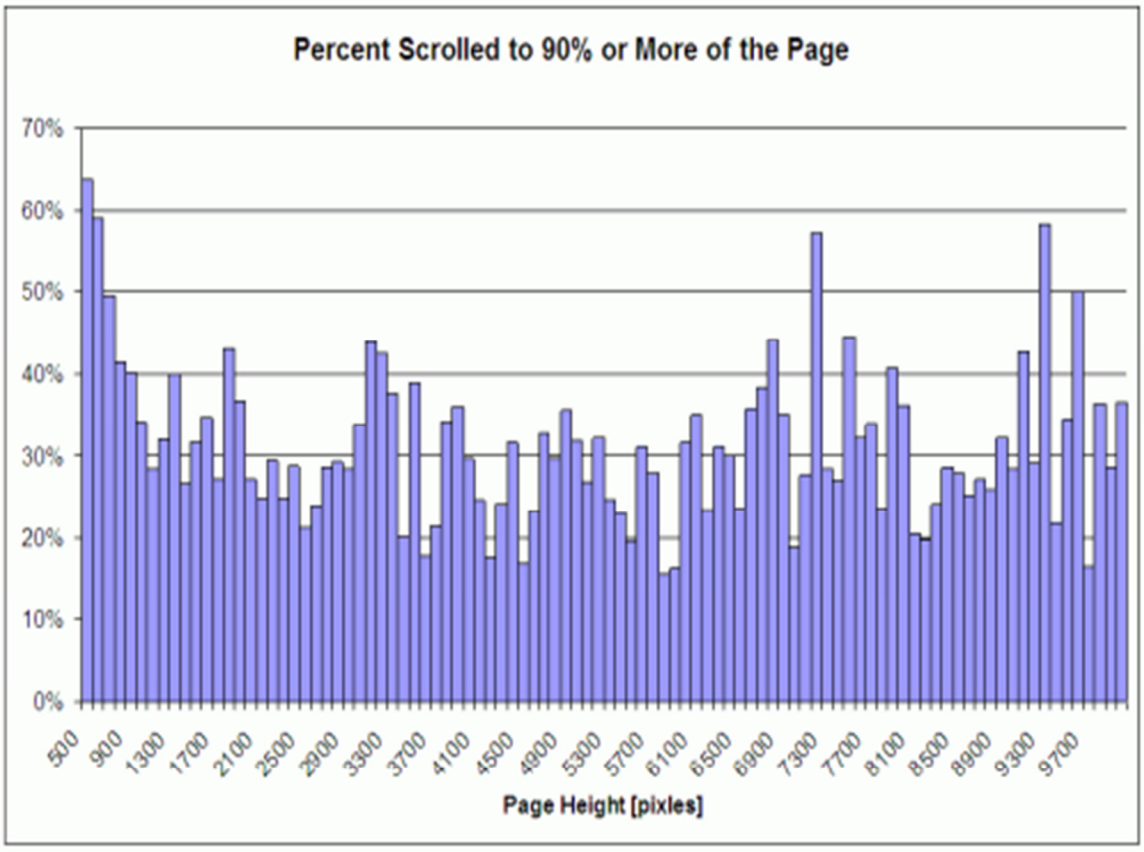

Fig. 2.5 Scrolling Percentages [2]

A study reported by Joe Leech of CX Partners, a user centered design agency, indicated that less content above the fold *even* encourages users to explore the content below the fold. The main point to take away here is that you shouldn’t stuff all your important content at the top because you fear that users won’t be able to find them otherwise.[3]

2.2.6 Place Important Content on the Left of a Web Page

People brought up in cultures where language is read and written from left to right have been trained early on in life to begin at the left of a page, whether in writing or reading a book. This can be the reason why many web users spend a majority of their attention on the left side of a web page — as much as 69% of the time, according to Dr. Nielsen's eye-tracking study that involved over 20 users.

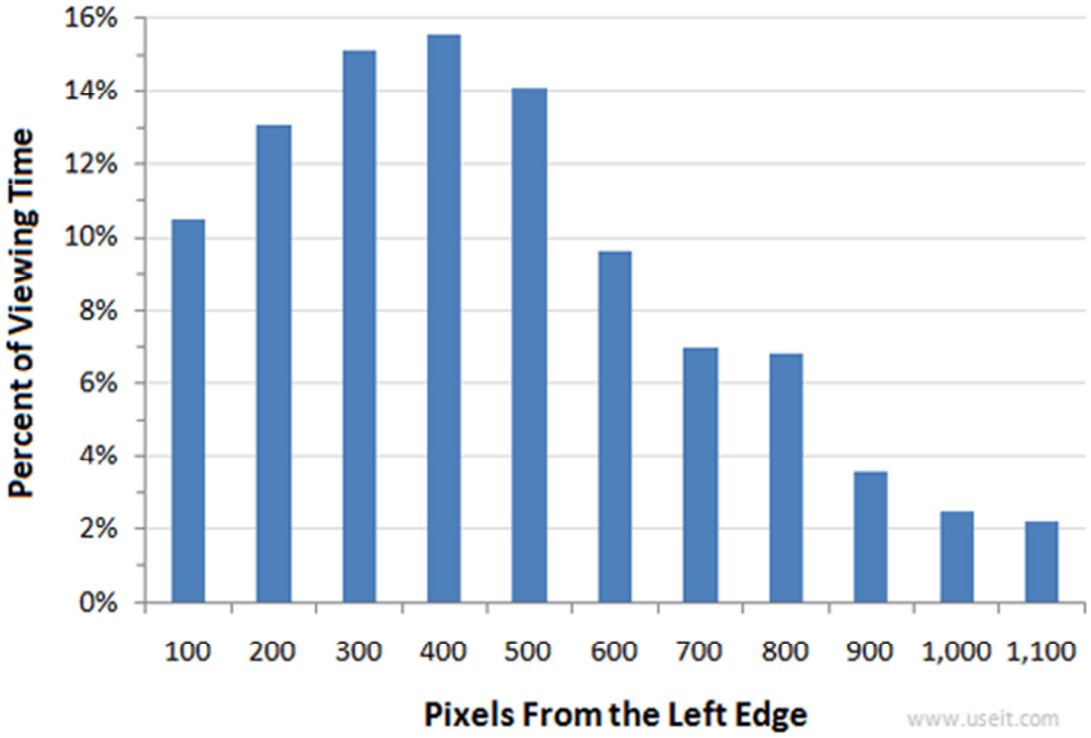

Fig 2.6 Effect of distance from left edge on viewing time [3]

The same results were reflected on websites whose language were read from right to left, such as Hebrew and Arabic sites, with the results inverted (higher attention on the right side versus the left).

There are two things to take away from this result. First, the language of your site matters when thinking about layout considerations; when designing websites you should consider cultural design considerations. Secondly, for sites that are traditionally read from left to right, placing important design components at the left is a good idea; vice versa for sites whose language is read from right to left.

2.2.7 Whitespace of Text Affects Readability

Easy readability of text improves comprehension and reading speed as well as enhancing the likelihood that a user will continue reading instead of abandoning the web page. There are many factors that influence ease of readability, including font choices (serif versus sans-serif), font-size, line-height, background/foreground contrast, as well as spacing.

A study on readability tested reading performance of 20 participants by presenting them with the same text blocks having different margins surrounding the text as well as varying line-heights (the distance between lines of text). It showed that text with no margins was read faster, however, reading comprehension decreased**.** [4]

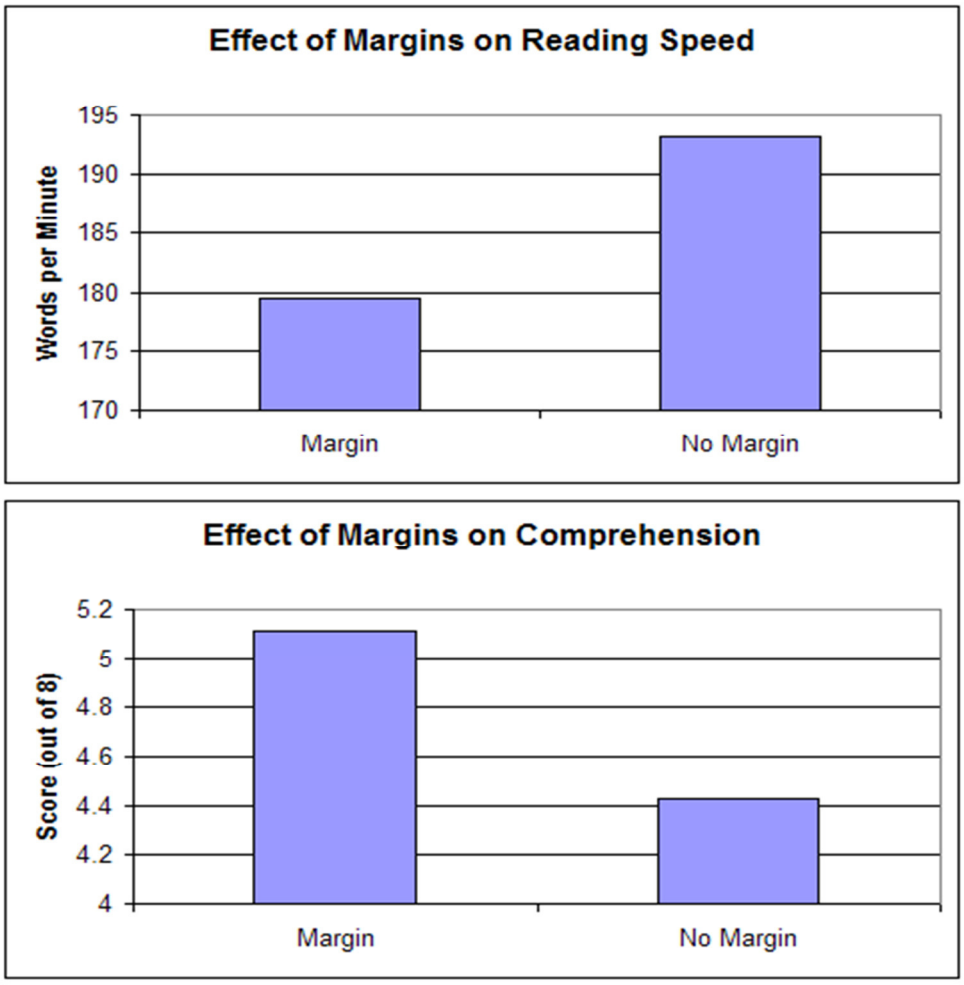

Fig. 2.7 Effect of margins on reading speed & comprehension [4]

As this particular study shows, the way we design our content can greatly impact the user's experience. Be wary of the details: color, line-heights, tracking, and so forth and be mindful of sound typography principles for the web to ensure that you're not discouraging your users from reading your content. Furthermore, study the effective use of negative space in web design.

2.2.8 Small Details Make a Huge Difference

Too often, we look at the big picture when creating a web design and ignore the little things when we're in a time crunch. We forego any thought put into the wording of something, or the design of a single button on a form if time and resources are limited. There are so many other things we need to think about that it's often easy to let go of the small stuff.

But something as small as a form's button can affect the success of a site, at least according to user interface design expert Jared Spool, who wrote about a case on how removing a button and adding a clear error message to avoid user errors in a checkout process increased revenue by $300 million in just a year. The first month witnessed a 45% additional sales attributed to the revision of the checkout process.

Pay attention to the details. Use A/B split testing to test your hypothesis and find out what is the most effective design that achieves better results. Set goals using analytics software to benchmark results of design tweaks in relation to site objectives. [4]

2.2.9 Don't Rely on Search as a Crutch to Bad Navigation

Users expect navigation to be easy to use and well organized. Even with an excellent site search engine, users will still turn to primary navigation first. According to a task test conducted by Gerry McGovern, over 70% of the participants began the task he gave them by clicking on a link on the page as opposed to using the search feature.

This result is similar to a test by UIE of 30 users that tracked e-commerce tasks. The research analysis concluded that "users often gravitated to the search engine when the links on the page didn't satisfy them in some way." Thus, search is most often utilized only when the user has failed to discover what they were looking for in the current page.

The lesson to be gained here is clear: Don't rely on site search to remedy poor content organization, findability issues, and bad information architecture. When users are unable to navigate to what they are looking for, attention should be diverted to layout, navigation, and content organization improvements, with improving search functionality as the secondary priority. [4]

2.3 API (Application Programming Interface)

An application programming interface (API) is a source code based specification intended to be used as an interface by software components to communicate with each other. An API may include specifications for routines, data structures, object classes, and variables.

An API specification can take many forms, including an International Standard such as Posix or vendor documentation such as the Microsoft Windows API, or the libraries of a programming language, e.g. Standard Template Library in C++ or Java API.

An API differs from an ABI (Application Binary Interface) in that the former is source code based while the latter is a binary interface. For instance POSIX is an API, while the Linux Standard Base is an ABI.)

An API can be:

- language-dependent, meaning it is only available by using the syntax and elements of a particular language, which makes the API more convenient to use.

- language-independent, written so that it can be called from several programming languages. This is a desirable feature for a service-oriented API that is not bound to a specific process or system and may be provided as remote procedure calls or web services. For example, a website that allows users to review local restaurants is able to layer their reviews over maps taken from Google Maps, because Google Maps has an API that facilitates this functionality. Google Maps' API controls what information a third-party site can use and how they can use it.

The term API may be used to refer to a complete interface, a single function, or even a set of APIs provided by an organization. Thus, the scope of meaning is usually determined by the context of usage.

In object-oriented languages, an API usually includes a description of a set of class definitions, with a set of behaviors associated with those classes. A *behavior* is the set of rules for how an object, derived from that class, will act in a given circumstance. This abstract concept is associated with the real functionalities exposed, or made available, by the classes that are implemented in terms methods (or more generally by all its public components hence all public methods, but also possibly including public field, constants, nested objects).

The API in this case can be conceived as the totality of all the methods publicly exposed by the classes (usually called the class *interface*). This means that the API prescribes the methods by which one interacts with/handles the objects derived from the class definitions.

More generally, one can see the API as the collection of all the *kinds* of objects one can derive from the class definitions, and their associated possible behaviors. Again: the use is mediated by the public methods, but in this interpretation, the methods are seen as a *technical detail* of how the behavior is implemented.

For instance: a class representing a `Stack` can simply expose publicly two methods `push()` (to add a new item to the stack), and `pop()` (to extract the last item, ideally placed on top of the stack).

In this case the API can be interpreted as the two methods `pop()` and `push()`, or, more generally, as the *idea* that one can use an item of type `Stack` that implements the behavior of a stack: a pile *exposing* its top to add/remove elements.

This concept can be carried to the point where a class interface in an API has no methods at all, but only behaviors associated with it. For instance, the Java language and Lisp (programming language)API include the interface `Serializable`, which requires that each class that implements it should behave in a serialized fashion. This does not require to have any public method, but rather requires that any class that implements it to have a representation that can be *saved* (serialized) at any time (this is typically true for any class containing simple data and no link to external resources, like an open connection to a file, a remote system, or an external device).

Similarly the behavior of an object in a concurrent (multi-threaded) environment is not necessarely determined by specific methods, belonging to the interface implemented, but still belongs to the API for that Class of objects, and should be described in the documentation.

In this sense, in object-oriented languages, the API defines a set of object behaviors, possibly mediated by a set of class methods.

In such languages, the API is still distributed as a library. For example, the Java language libraries include a set of APIs that are provided in the form of the JDK used by the developers to build new Java programs. The JDK includes the documentation of the API notation.

The quality of the documentation associated with an API is often a factor determining its success in terms of ease of use. [7]

2.4 MVC Architecture

Model–view–controller (MVC) is a software architecture currently considered an architectural pattern used in software engineering. The pattern isolates "domain logic" (the application logic for the user) from the user interface (input and presentation), permitting independent development, testing and maintenance of each (separation of concerns).

Model View Controller (MVC) pattern creates applications that separate the different aspects of the application (input logic, business logic, and UI logic), while providing a loose coupling between these elements.

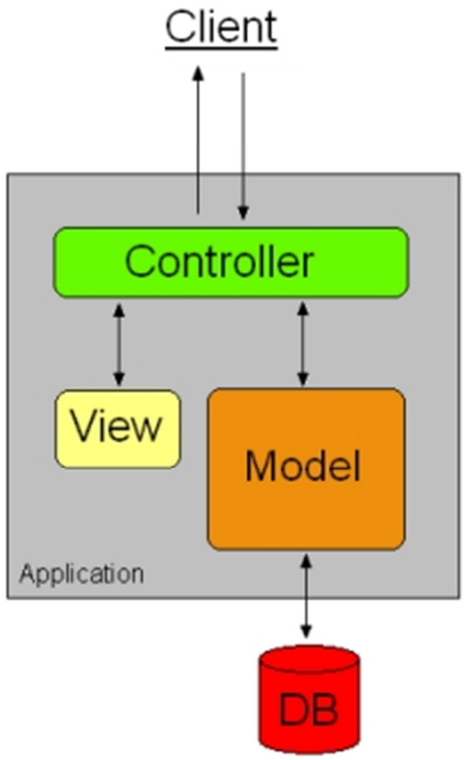

Fig. 2.8 MVC Architecture [7]

The model manages the behaviour and data of the application domain, responds to requests for information about its state (usually from the view), and responds to instructions to change state (usually from the controller). In event-driven systems, the model notifies observers (usually views) when the information changes so that they can react.

The **view** renders the model into a form suitable for interaction, typically a user interface element. Multiple views can exist for a single model for different purposes. A view port typically has a one to one correspondence with a display surface and knows how to render to it.

The controller receives user input and initiates a response by making calls on model objects. A controller accepts input from the user and instructs the model and a view port to perform actions based on that input.

MVC is often seen in web applications where the view is the HTML or XHTML generated by the app. The controller receives GET or POST input and decides what to do with it, handing over to domain objects (i.e. the model) that contain the business rules and know how to carry out specific tasks such as processing a new subscription, and which hand control to (X)HTML-generating components such as templating engines, XML pipelines, Ajax callbacks, etc.

The model is not necessarily merely a database; the 'model' in MVC is *both* the data and the business/domain logic needed to manipulate the data in the application. Many applications use a persistent storage mechanism such as a database to store data. MVC does not specifically mention the data access layer because it is understood to be underneath or encapsulated by the model. Models are not data access objects; however, in very simple apps that have little domain logic there is no real distinction to be made. Active Record is an accepted design pattern that merges domain logic and data access code — a model which knows how to persist itself [7]

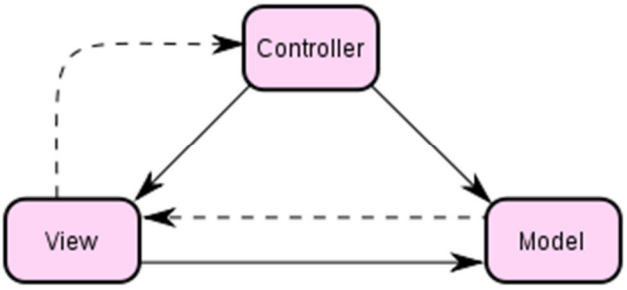

Fig. 2.9 Data Flow in MVC [7]

Chapter 3

System Development

3.1 Software Requirements Specification

3.1.1 Introduction

Users typically operate in a task-oriented mode. That is, a user has a set goal in mind (make an edit, set a preference, etc.). Such tasks may or may not require a user account (editing does not, but setting preferences does). In most cases, account creation is seen as one of the steps required to complete a specific task rather than a single task itself. Users want to get account creation out of the way and move on to what they were doing before the interruption. MediaWiki treats account creation (and login) as a single atomic task. In this mode, account creation is a "full stop" and it cannot be easily used for guidance. It is a distracting user experience, and one often forgotten or ignored.

3.1.2 Project Scope

- Excising Account Creation and Login Code from Special:UserLogin and putting it into its own class, making sure that Special:UserLogin still works so as not to break anything
- Converting Special:UserLogin to use HTMLForm for it's layout and display
- Adding "source" tracking and "configurable exit" functionality to the Creation/Login API
- Adding in the usability enhancements to the forms (into HTMLForm, where everyone can use them)
- Developing an AJAX library for Creation/Login
- Security Audit
- Documentation

3.1.3 Operating Environment

Software Interfaces :

1. Operating System: - Windows (Any Version)/Linux

2. Web browser: Internter Explorer (6 & above)/Firefox/Chrome/Safari/Opera

2. Development End(Programming Languages):- PHP, JavaScript, HTML, CSS

3. Database Server: - MySQL 5.5 Server

4. Web Server:-Apache Server.

Hardware Interfaces:

1. Processor:-PIV- 500 MHz to 3.0 GHz.

2. RAM: - 256MB.

3. Disk: - 10 GB.

4. Monitor: - 15” inch.

5. Standard Keyboard and Mouse.

6. Internet connection (Broadband/Dial-up).

3.1.4 System Features

Signup API

This module acts as the controller for handling third party requests for new account creation

Given Input:

All details of the registering user i.e username, password, retyped password, email address, real name

Expected Output:

On successful account creation, this module will return a user id for the new user & a login token

Source Tracking

This module tracks the events that lead to new user registrations. It helps in gaining insights on which articles & actions are the most influential in user conversion

Given Input:

All details about the source event such as article namespace & action

Expected Output:

On successful registration this module will store the source event in the database

Internationalization

This module helps in making the extension accessible in various languages

Given Input:

The message to be displayed in English & the desired output langauge

Expected Output:

The corresponding translated message in the desired language

Hooks

This module helps in making the application extensible by third party developers, allowing them to add their own methods to be executed at specified points in the application

Given Input:

The function name & event after which it should be executed

Expected Output:

The function runs after the specified event

3.1.5 Non Functional Requirements

3.1.5.1 Performance requirements

Wikipedia has a database of more than 16 TB & hence the following the following performance features are necessary

- The application must allow concurrent access by thousands of users
- Database queries must be well optimized for fast execution
- Application should consume less amount of main memory
- Maximize use of client & server side caching

3.1.5.2 Security requirements

- The application must prevent spam attacks
- Application must be able to prevent CSRF, SQL Injection & other security vulnerabilities

3.2 Analysis Model

3.2.1 Class Diagram

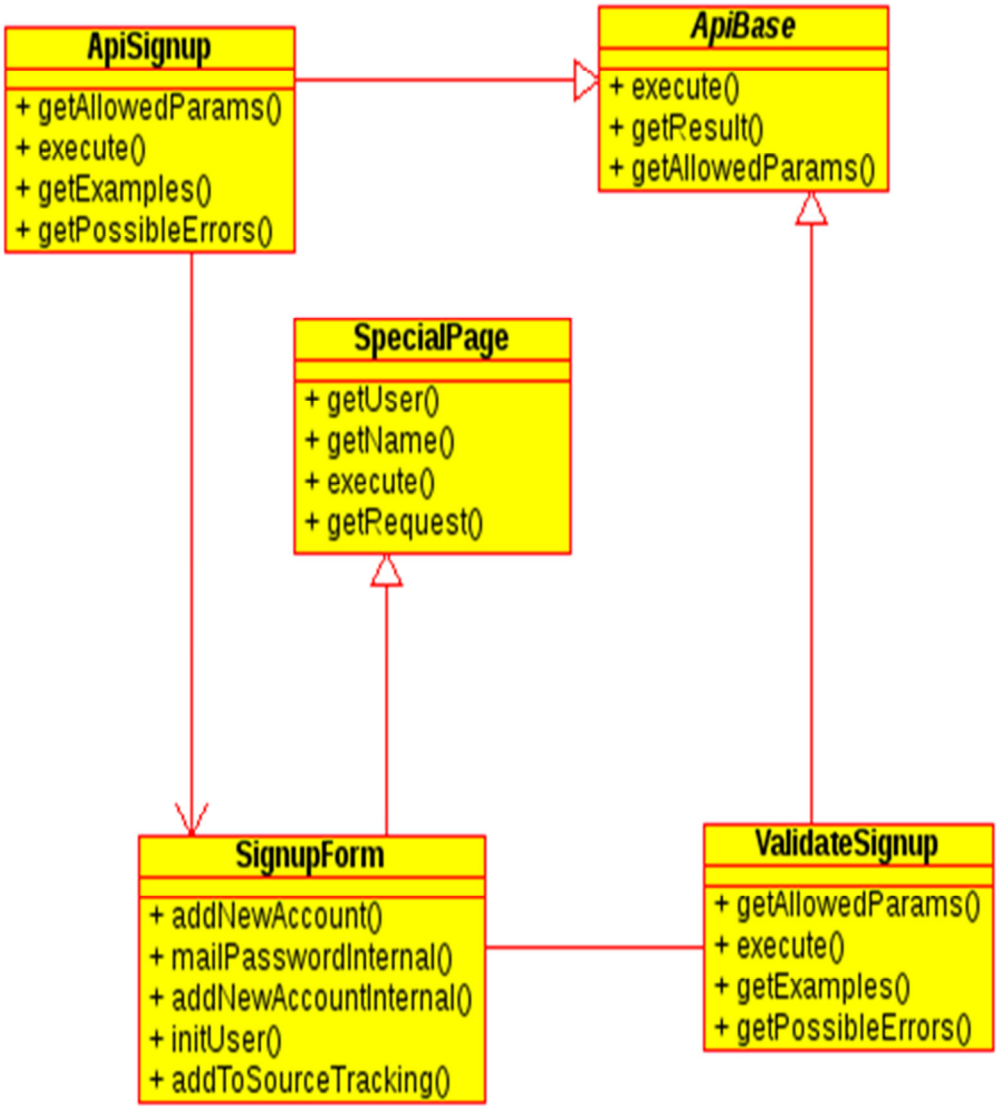

Fig. 3.1 Class Diagram

3. 3 System Design

3.3.1 Architecture

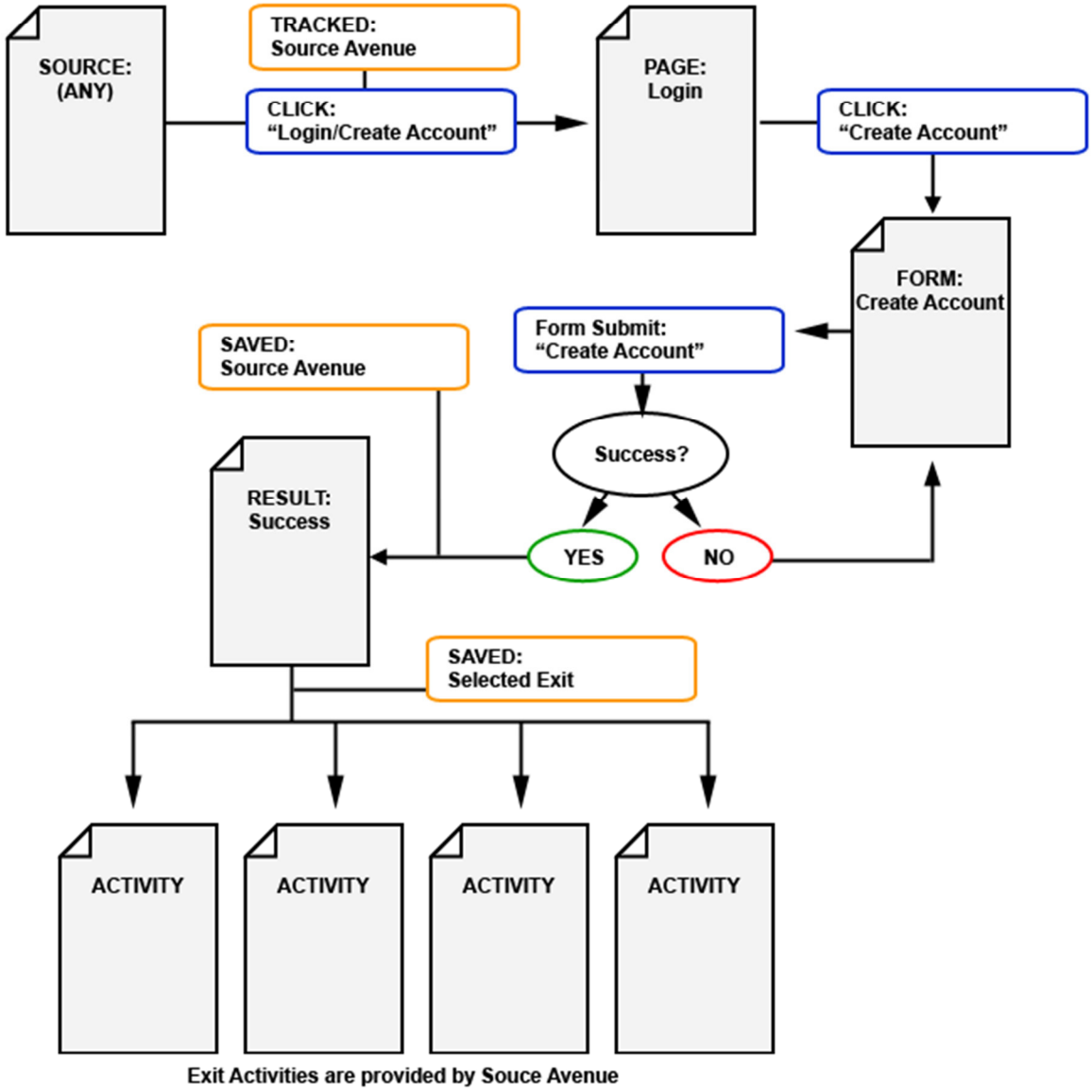

Fig. 3.2 System Architecture

3.3.2 UML Diagrams

3.3.2.1 Use Case Diagram

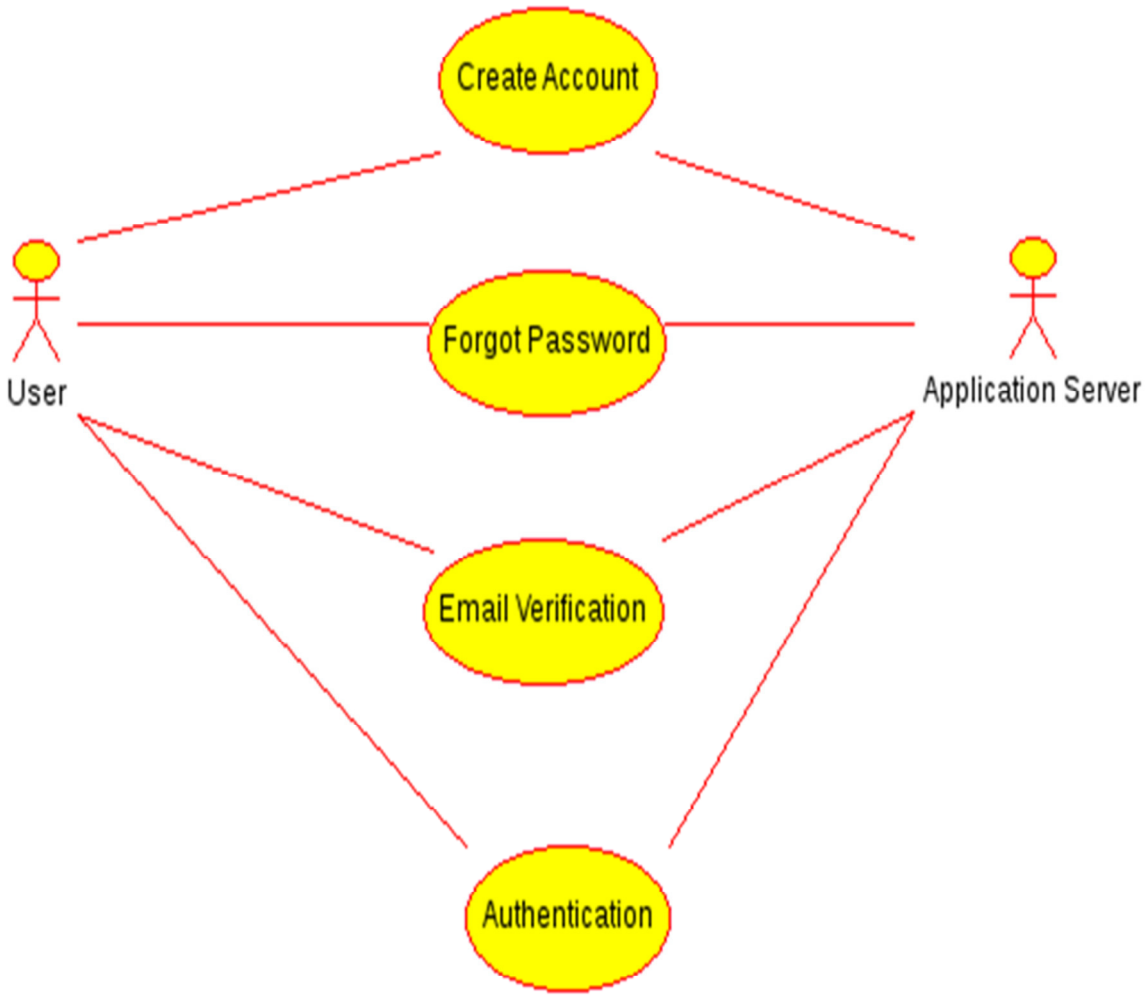

Fig. 3.3 Use Case Diagram

3.3.2.2 Sequence Diagram

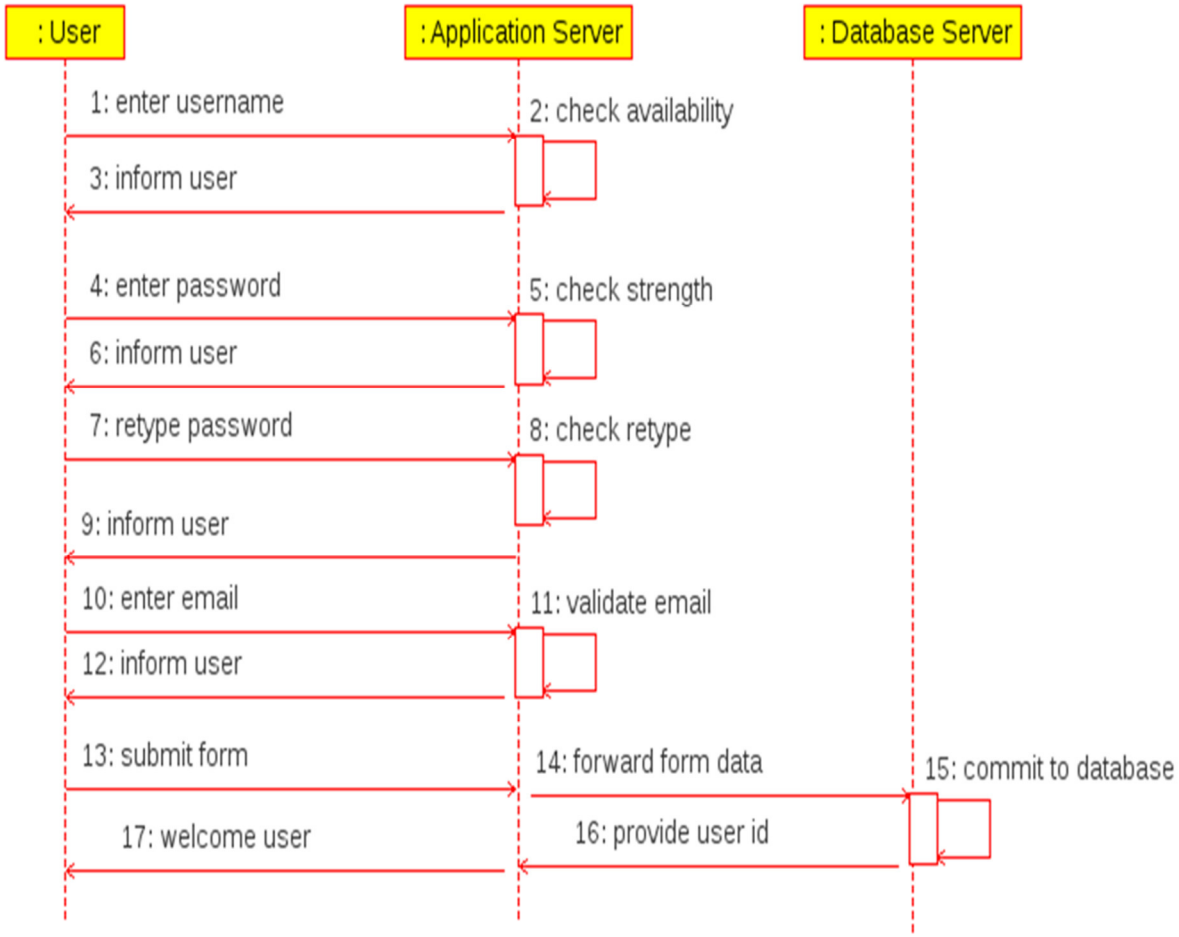

Fig. 3.4 Sequence Diagram

3.4 Technical Specification

3.4.1 PHP

PHP is a general-purpose scripting language that is especially suited to server-side web development where PHP generally runs on a web server. Any PHP code in a requested file is executed by the PHP runtime, usually to create dynamic web page content or dynamic images used on web sites or elsewhere. It can also be used for command-line scripting and client-side GUI applications. PHP can be deployed on most web servers, many operating systems and platforms, and can be used with many relational database management systems (RDBMS). It is available free of charge, and the PHP Group provides the complete source code for users to build, customize and extend for their own use.

PHP acts primarily as a filter, taking input from a file or stream containing text and/or PHP instructions and outputting another stream of data; most commonly the output will be HTML. Since PHP 4, the PHP parser compiles input to produce bytecode for processing by the Zend Engine, giving improved performance over its interpreter predecessor

Originally designed to create dynamic web pages, PHP now focuses mainly on server-side scripting and it is similar to other server-side scripting languages that provide dynamic content from a web server to a client, such as Microsoft's ASP.NET, Sun Microsystems' JavaServer Pages and mod_perl. PHP has also attracted the development of many frameworks that provide building blocks and a design structure to promote rapid application development (RAD). Some of these include CakePHP, Symfony, CodeIgniter, and Zend Framework, offering features similar to other web application frameworks.

The LAMP architecture has become popular in the web industry as a way of deploying web applications. PHPis commonly used as the *P* in this bundle alongside Linux, Apache and MySQL, although the *P* may also refer to Python or Perl or some combination of the three. Similar packages are also available for Windows and Mac OS X, then called WAMP and MAMP, with the first letter standing for the respective operating system.

As of April 2007, over 20 million Internet domains had web services hosted on servers with PHP installed and mod_php was recorded as the most popular Apache HTTP Server module. PHP is used as the server-side programming language on 75% of all web servers. Web content management systems written in PHP include MediaWiki, Joomla, eZ Publish, WordPress, Drupal and Moodle. All websites created using these tools are written in PHP, including the user-facing portion of Wikipedia, Facebook

3.4.2 MySQL

MySQL is a relational database management system (RDBMS) that runs as a server providing multi-user access to a number of databases. It is named after developer Michael Widenius' daughter. The SQL phrase stands for Structured Query Language.

The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements. MySQL was owned and sponsored by a single for-profit firm, the Swedish company MySQL AB, now owned by Oracle Corporation

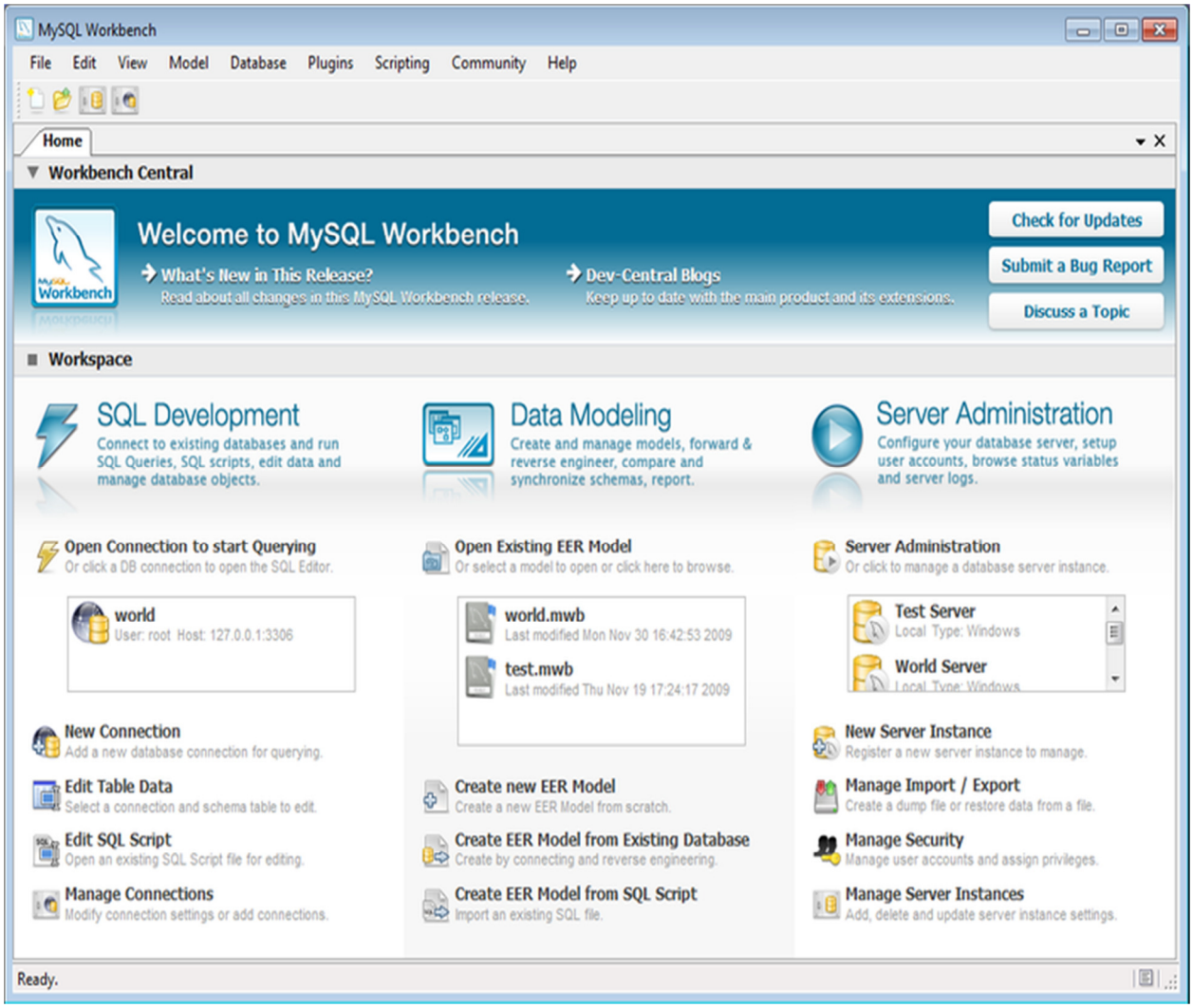

Fig. 3.5 MySQL Workbench

3.4.3 Ajax

Ajax is a group of interrelated web development methods used on the client-side to create asynchronousweb applications. With Ajax, web applications can send data to, and retrieve data from, a server asynchronously (in the background) without interfering with the display and behavior of the existing page. Data is usually retrieved using the XMLHttpRequest object. Despite the name, the use of XML is not needed (JSON is often used instead), and the requests do not need to be asynchronous.

Ajax is not one technology, but a group of technologies. HTML and CSS can be used in combination to mark up and style information. The DOM is accessed with JavaScript to dynamically display, and to allow the user to interact with the information presented. JavaScript and the XMLHttpRequest object provide a method for exchanging data asynchronously between browser and server to avoid full page reloads.

The term *Ajax* has come to represent a broad group of web technologies that can be used to implement a web application that communicates with a server in the background, without interfering with the current state of the page. In the article that coined the term Ajax, Jesse James Garrett explained that the following technologies are incorporated:

- HTML (or XHTML) and CSS for presentation
- The Document Object Model (DOM) for dynamic display of and interaction with data
- XML for the interchange of data, and XSLT for its manipulation
- The XMLHttpRequest object for asynchronous communication
- JavaScript to bring these technologies together

Since then, however, there have been a number of developments in the technologies used in an Ajax application, and the definition of the term Ajax. In particular, it has been noted that JavaScript is not the only client-side scripting language that can be used for implementing an Ajax application; other languages such as VBScript are also capable of the required functionality. JavaScript is the most popular language for Ajax programming due to its inclusion in and compatibility with the majority of modern web browsers. Also, XML is not required for data interchange and therefore XSLT is not required for the manipulation of data. JavaScript Object Notation (JSON) is often used as an alternative format for data interchange although other formats such as preformatted HTML or plain text can also be used.

3.5 Code Snippets

3.5.1 Installation

To install this extension, copy the entire SignupAPI directory to your extensions directory

Add the following to LocalSettings.php:

Note: $IP stands for the root directory of your MediaWiki installation, the same directory that holds LocalSettings.php.

```
require_once("$IP/extensions/SignupAPI/SignupAPI.php");
$wgUseAjax = true;
$wgSignupAPIUseAjax = true;
$wgSignupAPISourceCreation = true;
$wgUseCombinedLoginLink = false;
```

Finally, take a backup of your existing database & run update.php to create the sourcetracking table.

Configuration parameters

$wgUseAjax This paramater is required to use AJAX & is set to true by default in MediaWiki 1.17 & above

$wgSignupAPIUseAjax If set to true, the extension will AJAX-ify the signup form & facilitate the validation of username, password, retype & email dynamically before submitting the form

$wgSignupAPISourceCreation If set to true, the extension will add the source tracking paramaters to the 'Create account' link & on successful signup will store them inside the sourcetracking table

$wgUseCombinedLoginLink This parameter is used to display separate links for Login & Create account instead of the combined 'Login/Create account' link.

3.5.2 Internationalisation

```
$messages = array();

//English
$messages['en'] = array(
      'usersignup' => 'User signup',
      'signupapi-desc'  =>  'Cleans  up  the  [[Special:UserLogin|login
page]] from signup related stuff and adds an API for signup',
      'signupapi-ok' => 'OK',
      'signupapi-noname' => 'No username was specified',
      'signupapi-userexists' => 'User exists',
      'signupapi-enterpassword' => 'You must enter a password',
      'signupapi-passwordtooshort' => 'Password is too short',
      'signupapi-weak' => 'Weak',
      'signupapi-medium' => 'Medium',
      'signupapi-strong' => 'Strong',
      'signupapi-badretype'  =>  'The  passwords  you  entered  do  not
match',
      'signupapi-passwordsmatch' => 'Passwords match',
      'signupapi-invalidemailaddress' => 'E-mail address is invalid',
      'signupapi-invalidusername' => 'Username entered is invalid',
);

// Arabic
$messages['ar'] = array(
      'signupapi-weak' => 'ضعيف',
      'signupapi-medium' => 'متوسط',
);

//Serbian
$messages['dsb'] = array(
      'usersignup' => 'Wužywarske registrěrowanje',
      'signupapi-desc'  =>  'Rumujo  [[Special:UserLogin|psizjawjeński’,
      'signupapi-ok' => 'W póměźe',
      'signupapi-noname' => 'Wužywarske mě njejo se pódało',
      'signupapi-userexists' => 'Wužywań eksistěrujo',
      'signupapi-enterpassword' => 'Musyš gronidło zapódaś',
      'signupapi-passwordtooshort' => 'Gronidło jo pśekrotke',
      'signupapi-weak' => 'Słabe',
      'signupapi-medium' => 'Srědne',
      'signupapi-strong' => 'Mócne',
      'signupapi-badretype' => 'Gronidle, kótarejž sy zapódał, se
};
```

3.5.3 Form Validation

```
function validateInput( fieldtype, fieldid ) {
          var inputVal = document.getElementById( fieldid ).value;
          var valresult = document.getElementById( fieldid + 'val' );
          $.ajax({
               type: "POST",
               url: mw.util.wikiScript('api'),
               data:  {  'action':'validatesignup',  'format':'json',
'field':fieldtype, 'inputVal':inputVal },
               dataType: 'json',
               success: function( jsondata ){
                    var   image   =   "<img   src='"+   imagePath   +
jsondata.signup.icon +"'>";
                    var message = jsondata.signup.result;
                    valresult.innerHTML = image + message;
               }
          });
     }
```

3.5.4 Hooks

```
class SignupAPIHooks {

     /**
      * @param $updater DatabaseUpdater
      * @return bool
      */
     static function onSourceTracking( $updater = null ) {
          if ( $updater !== null ) {
               $base = dirname( dirname( __FILE__ ) );
               $updater->addExtensionUpdate(    array(    'addTable',
'sourcetracking',
                    "$base/sourcetracking.sql", true ) );
          } else {
               global $wgExtNewTables;

               $wgExtNewTables[] = array(
                    'sourcetracking',
                    'sourcetracking.sql'
               );
          }
          return true;
     }
}
```

3.6 Final Outcome

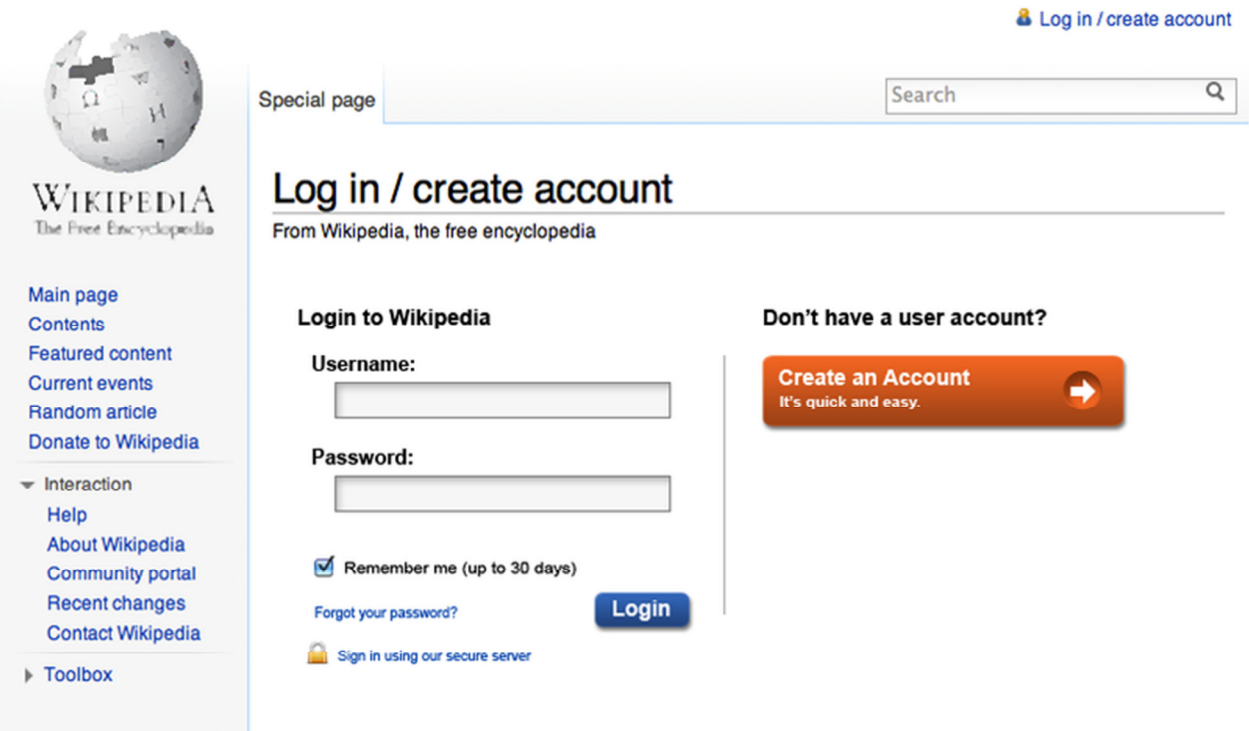

Fig. 3.6 New Login Screen

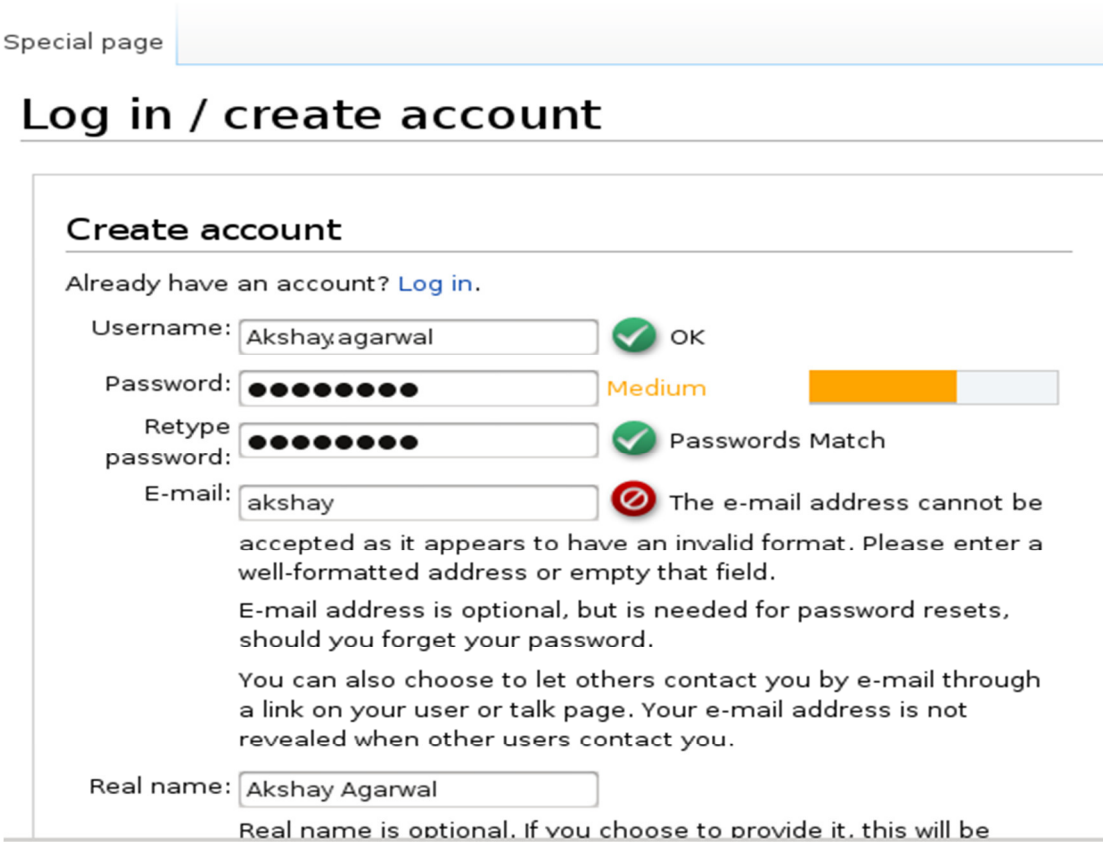

Fig. 3.7 New Account Creation Screen

Chapter 4

Performance Analysis

4.1 A/B-testing

The following diagram shows the elements which were A/B tested:

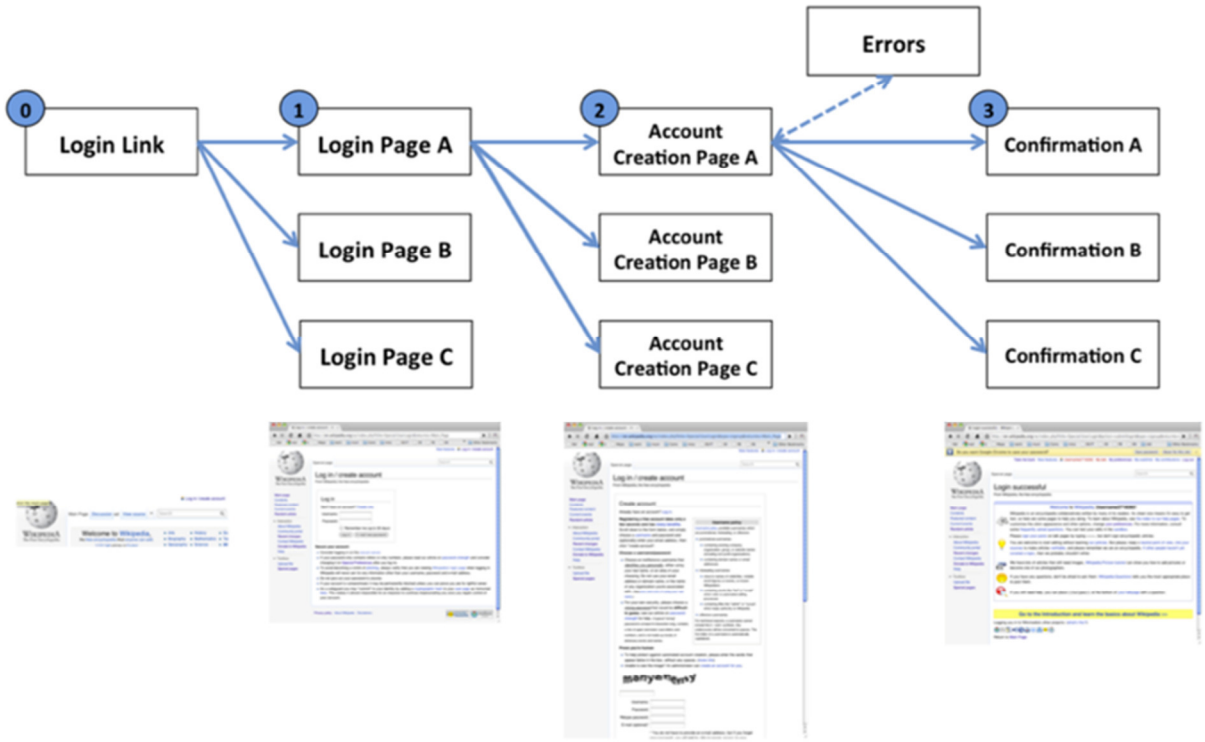

Fig. 4.1 A/B Testing

Each of the three main steps within the flow will have multiple versions. Examples of each of these versions are included.

4.1.1 Requirements for A/B testing

1. Ability to implement different pages for each step in the flow
2. Ability to set percentages for each of the pages (% of users that see a given page)
3. Ability to track fallout of each flow

4.1.2 Analytics Requirements

- Impact on editing: The first priority is to understand whether different versions of Page 3 result in different editing behavior. We need to be able to track which version users see so that we can run editing histories against users which have seen different versions of the page.
- We do not need to analyze the effects of different combinations of login-account creation-confirmation on editing. Simply measuring the effect of different versions of the Confirmation page is sufficient
- Funnel analysis: In order to understand the effectiveness of each point in the flow, we would ideally have funnel analysis (% of users that see 1a, 2a, 3a, etc.).
- An approximation for this is to be able to report the click-through-rates for each page

4.1.3 Test settings management

There needs to be a configurable way to manage various 'treatment plans' and rates of users to get them. At the very simplest, a global configuration arrays that looks like this:

```
$wgBuckets = array( array( "name" => "A", "rate" => .5),
array( "name" => "B", "rate" => .5));
```

Whether it is used will depend on whether or not we want to treat the steps as independent or not.

4.1.4 Test Cases

A/B-testing in step 1 (Login)

On page load, the software checks if the user has a "bucket" cookie set. If not, user will need to be given a bucket cookie that corresponds to the bucket they're in. Based on this, the user will be sent to the appropriate login page. This will need to send information to the data collection mechanism.

A/B-testing in step 2 (Account creation page)

On page load, the software needs to perform the same check as above, and be sent to the appropriate account creation page, and this will need to be sent to the data collection mechanism.

A/B-testing in step 3 (Confirmation)

Use of the function "UserLoginComplete" in "Specialuserlogin.php": the developer needs to deploy an extension that provides a switcher which switches based on the configuration settings in the function "successfulCreation()". This switcher will send people either to the existing page "MediaWiki:Welcomecreation" or to a new page "MediaWiki:WelcomecreationAlternative". MediaWiki:Welcomecreation is the existing page that is shown to all users who create an account. Every admin on a local Wikipedia will be able to change it. For the bigger language versions of Wikipedia it will be sufficient if 20% of the new users will be directed to the new page MediaWiki:WelcomecreationAlternative that we use for our testing purposes.

4.1.5 Results

The results in the Account Creation Improvement Project were collected in a rather lengthy report. Here are the highlights:

- the surveys showed us that people wanted to be "a part of Wikipedia"
- here are some of the "before" results on various language versions
- below are the results from the first low-quality tests that we ran from February 23-March 30
- the high-quality tests that we ran, showed a very clear "win" for this version of the account creation processes
- when using that version of the account creation process, we can increase the number of newcomers who make 1 edit with 15% on an average day
- that version of the account creation process also increases the number of newcomers who make 5 edits with about 1%. These figures may seem very abstract, but by increasing the inflow of new Wikipedians who make 5 edits from 6.12% to 7.02 on an average day, we gain 9 persons per day! From 59.90 (with the present model) to 68.90 persons (with ACP1).

English Wikipedia Results for February 2012

Day	Number of new user accounts	New accounts with at least one edit on the day of account creation	Percentage
1	7,132	1,997	28%
2	6,980	1,990	29%
3	6,922	1,949	28%
4	6,894	1,991	29%

5	6,035	1,703	28%
6	6,410	1,761	27%
7	7,122	1,969	28%
8	6,855	1,825	27%
9	7,283	2,099	29%
10	6,804	2,020	30%
11	6,638	1,865	28%
12	5,926	1,599	27%
13	6,348	1,754	28%
14	6,963	1,983	28%
15	7,732	2,086	27%
16	7,380	2,114	29%
17	7,180	2,177	30%
18	6,561	1,914	29%

19	5,906	1,712	29%
20	6,068	1,681	28%
21	7,112	2,057	29%
22	7,201	2,064	29%
23	7,341	2,151	33%
24	7,099	2,063	29%
25	6,574	2,328	35%
26	5,938	1,614	27%
27	6,197	1,710	27%
28	7,279	2,192	30%

Chapter 5

Conclusions

5.1 Conclusion

This project is helping Wikipedia in offering a better user experience & in obtaining in depth analysis which will ultimately lead to creation of more & better articles. It will also help in providing more free knowledge to the world

5.2 Advantages

- The user interface for account creation becomes highly intuitive
- It saves time of millions of new users
- Source tracking helps in better user conversion
- Improved analytics
- It can be extended by third party extensions using API
- Due to internationalisation it can be used by users all over the world in their own native languages

5.3 Disadvantages

- Existing account flow structure needs to be modified
- Extensions need to be modified for supplying exit activities

5.4 Future Scope

- This project can be extended to integrate with several other extensions such as ClickTracking which can help in providing an enhanced analytics.
- Third Party extensions can be modified to supply suitable exit activities for every user after account creation

5.5 Usage Examples

5.5.1 General Account Creation

An anonymous user decides to create an account in order to "be part of Wikipedia." They click the "Login/Create Account" link while viewing the article, "Roman Mythology."

The source of the account creation, "Page: Roman Mythology" is tracked. This is a "general" account creation and does not have an obvious task associated with it.

The user fills out the account creation form and the process is completed successfully. At this point, Special:UserLogin informs the account API that the following activities are possible exits:

- Browse other articles in the "Mythology" category
- Edit or discuss the "Romany Mythology" page
- Go to the User Preferences page
- View a tutorial for new editors
- Sign up for a mentorship program
- Edit a page in the Sandbox
- Return to the "Roman Mythology" page

The activity chosen is also tracked. From this, in the future, we can determine:

- Which tasks users are most likely to perform upon successful account creation, and
- Which immediate tasks are most likely to result in user conversion (e.g., "users who view the tutorial are 20% more likely to convert than those who do not"), which will allow us to better tailor exit activities towards conversion

5.5.2 LiquidThreads Posting (AJAX)

An anonymous user writes a posting on a LiquidThreads page. Upon clicking the "submit" button, he or she is informed that they are posting anonymously and given three options in-situ via Javascript dialogs:

- Post anonymously
- Login
- Create Account

Logging in or creating an account would be handled through an AJAX dialog. The user opts to create an account.

LiquidThreads informs the Account API that the source avenue is "LiquidThreads Posting" and also informs the API that possible exit activities should include:

- Submit Posting
- Abort Posting
- Set LiquidThreads Preferences
- Open user preferences (in a new window)

If the account creation is successful, the user's authentication credentials are updated, and the dialog is changed to show a successful account creation and provides the additional exit options.

The process of account creation (or login) does not wildly interrupt the user's task flow.

REFERENCES

[1] Norman, Donald A. and Nielsen, Jakob (2010): Gestural interfaces: a step backward in usability. In Interactions, 17 (5) pp. 46-49.

[2] Nielsen, Jakob (2006): Progressive Disclosure.
Available from <http://www.useit.com/alertbox/progressive-disclosure.html>

[3] Vredenburg, Karel, Mao, Ji-Ye, Smith, Paul W. and Carey, Tom (2002): A survey of user-centered design practice. In:Terveen, Loren (ed.) Proceedings of the ACM CHI 2002 Conference on Human Factors in Computing Systems Conference April 20-25, 2002, Minneapolis, Minnesota. pp. 471-478.

[4] Dow, Steven P., Glassco, Alana, Kass, Jonathan, Schwarz, Melissa, Schwartz, Daniel L. and Klemmer, Scott R. (2010): Parallel prototyping leads to better design results, more divergence, and increased self-efficacy. In ACM Transactions on Computer-Human Interaction, 17 (4) p. 18

[4] User Centric Design, U.S. Department of Health and Human Services.
Available from: <http://www.usability.gov/>

[5] Account Creation Improvement Project.
Available from <http://www.mediawiki.org/wiki/Account_Creation_Improvement_Project>

[6] Wikimedia Engineering Report, March 2011.
Available from<http://www.mediawiki.org/Wikimedia_engineering_report/2011/March>

[7] MediaWiki Developer Reference.
Available from <http://www.mediawiki.org/wiki/Manual:Contents>