Header

Title Page

Comment anything requirements Document.

Instructor Comments and Evaluation

Table of Contents

Abstract

We describe the results of Software Requirement Engineering in a Waterfall model as applied to our proposed Software Product, Comment Anywhere, a browser extension for commenting on webpages. In this document, the requirements of the product are developed. We begin by addressing the background, objective, and team details. We then explore the Application Domain, Operating Requirements, and describe Data Sources and Use Cases. Finally, we explicate the functional, nonfunctional, and documentary requirements of Comment Anywhere.

Introduction

Background

Internet denizens have long found ways to have vibrant communications about a wide variety of content. In the past, more website supported these conversations through comment sections, but many have closed their comments in recent years. [2] Instead, the avenues of discourse have become social media sites such as Facebook, Reddit, and bulletin board style forums, decoupling the conversation from the content itself.

Tying comments to social media posts rather than the content has the effect of fragmenting the conversation and diluting information available to viewers. Evidence exists that users spend less time on webpages without comments and report a worse user experience. [3] We infer that there is a market for comments tied to content rather than posts and an application that can bring comments to the content could capture monetizable user engagement.

Objective

The objective of this project is to create a web based browser extention which allows people to comment on websites based on their url. This is useful for websites that dont have comment sections, especially when viewers have additional thoughts or information to offer. We will have typical features of commenting systems, including reporting, automated moderation, manual moderation, and account control and additional unique features. Our aim is to launch the product as a profitable business, and the goal of this project is to meet Phase 1 of our business plan.

\*\* Phase 1: Initial Rollout\*\*

* Less than 50,000 monthly users
* Goal is to garner interest and goodwill
* Engage with users, listen to feedback, implement feature requests

\*\* Phase 2: "Buy Me a Coffee"\*\*

* Less than 200,000 monthl users
* Continue to build interest and goodwill
* Form corporation / LLC
* Capitalize on goodwill via donations to offset costs

\*\* Phase 3: "Good Place to Advertise"\*\*

* Roll out ad purchasing system
* Intersperse ads with comments on relevant content similar to google search results or reddit comments
* Ad Sales teams
* Scale

We hope to create a product that is able to hit the market and become the new way to comment on the internet free from those that seek to stop the free flow of information.

Team Details & Dynamics

Our team is git-committed and document driven with an eye to the business plan.

Our development strategy is one of flexibile independence that allows all members to modify all code and prose, without clearing everything all the time. The policy is to merge all pull requests if they have no conflicts and pass integration tests. At this stage, it is better to roll back a merge than to stagnate. Just like our Product, Comment Anywhere, team members have freedom and the responsibility that comes with it. There can be no excessive emotional ownership of code or prose; we must change and improve each other's work, we must proofread and unit test.

To ensure all our talents are brought to bear we have to bring each other up to speed. We will make effort to assist each other in the learning of new technologies. We will comment in and on our work constantly, creating guides for using and developing our features.

Every group member has committed to being the Team Leader for a portion of the project.

| **Team Leader** | **Phase** |
| --- | --- |
| Robert Krency | Requirements |
| Frank Bedekovich | Analysis |
| Karl Miller | Design |
| Luke Bates | Implementation |

Application Domain

The application domain of Comment Anywhere is internet communication services. More specifically, the domain is internet users commenting and viewing comments about web pages.

Initial Business Model

Operating Environment

The Front End will run in the browser engines of Chrome, Firefox, and other browsers that may support Chromium or Firefox based extensions. The Back End will be configured to run at least two Virtual Machines to allow for simple deployment on a variety of cloud options. The Database and HTTP Server will run on separate Virtual Machines.

Description of Data Sources

We will be getting many parts of our data from many different sources, first we will grab the url that the user is currently on and request the data for that webpages comment section. We will then see that request and grab the comment section from our database and display them to the user. When the user is ready, we will have the user log in to the extention. After the user enters their information into the extention we will verify if the user is actually the user as with all logins. There should be a few options for the user at this point the user can choose to either make a comment or reply to another comment, in the moderator's case they will also be able to report comments. If the user comments or reply's when the commenter posts their message to the database for the website url. If a moderator reports a comment the comment will move from the comments database to a removed comments database that only the moderators are able to see. The administrators as well as being able to do everything users and moderators can do is approve and select both global and regular moderators, they can also request and see who has approved the moderators and when for both global and local moderators. Global moderators are also able to request and see local moderators approval and time. Users can also be banned by moderators, regular moderators cana only ban users to the domain of the site they moderate while global and administrators can issue domain and full bans from all sites. When a user is banned their name is put into the domain bans database as well as where they were banned from who banned them and when.

Use Case Diagrams

// inheritance\_diagram.png

User privileges may be represented by this inheritance diagram. For instance, *members* inherit all *guest* abilities.

// guest\_diagram.png

*Guests* do not need an account to view comments. However, they cannot interact with our services without logging into a registered account.

// member\_diagram.png

*Members* may interact with our service by posting and responding to comments. Users may also report comments, view hidden comments, and report bugs.

// domain\_mod\_diagram.png

Domain moderators manage one or more assigned domains. They may access and remove reported comments, recategorize comments, and ban members. These actions are restricted to the assigned domain. // global\_mod\_diagram.png

Global moderators manage all domains. Additionally, they can remove domain moderators.

// admin\_diagram.png

Administrators manage all domains, remove global moderators, and view internal logs and metrics.

Initial Requirements

Functional Requirements

Comment Anywhere has a number of functional requirements necessary to meet our objective.

* Getting Comments

A Viewer must be able to request comments from the Server by clicking the extension icon in their browser. The Server must be able to serve the User comments related to that URL. The Browser Extension must be able to display those comments to the User.

Logged In Users may also receive comments that are hidden by default to non-logged in Viewers.

* Registering

A Viewer must be able to register a new account from the user interface in the drop down portion of the browser extension. The Server must be able to validate that the User does not already exist and that their password is of sufficient strength, then either add that User to the Database or tell the User their was a problem with registration.

* Logging In

The User must be able to log into an account from the user interface in the drop down portion of the browser extension. The Server must verify whether the User has supplied the correct credentials. The Server must be able to track whether an HTTP Request is coming from a logged in user.

* Settings

The User must be able to change their settings locally to control whether they want to view potentially problematic, hidden, comments. They must be able to reset their password from their settings page.

* Posting Comments

A logged-in User must be able to post a new comment. The Server must be able to add that Comment to the comment data for the URL the User is commenting on, if the user is permitted to comment on that page.

* Validating Comments and Usernames

The Server must automatically evaluate Comments and Usernames to determine if they may contain words or phrases which violate our policies. They must prevent and remove comments that certainly violate the policies and flag and hide comments that may do so. It must prohibit the registration of usernames that contain prohibited words.

* Reporting

A logged-in User must be able to report a rule-breaking comment. The Server must be able to track which comments require moderation action. A logged-in User must also be able to report buggy pages.

* Moderating

A Domain Moderator or Global Moderator must be able to view comments that have been flagged by Users as rule breaking. They must be able to remove rule breaking comments, take other actions, or clear flags if no action is required.

* Assigning Moderators

A Global Moderator must be able to elevate a User to the status of Domain Moderator. An Admin must be able to assign Global Moderators. Similarily, an Admin must be able to remove Global Moderator permissions and Admins and Global Moderators must be able to remove Domain Moderator Permissions. The Server must be able to accurately evaluate permissions and allow or disallow actions based on those permissions.

* Banning

Domain Moderators must be able to ban troublesome Users from their domain. The Server must not allow banned Users to post on that domain. Global Moderators must be able to ban Users globally or from a specific domain. Users must be able to appeal Bans and Admins must be able to review banning actions taken by Moderators.

* Reports

Admins must be able to view reports on User Activity, Moderation Actions, and other metrics.

* Static Website

There must be a website that provides a description of the project, download links for the Browser Extension, and instructions for installing and using the Browser Extension, which anyone on the internet can view.

NonFunctional Requirements

* Memory and Bandwidth Usage

The Server and Database must be optimized to minimize Memory and Bandwidth Usage as much as possible. We will likely be paying for cloud hosting and have no source of income throughout the Phase 1 roll out. Therefore, we will need to optimize our Server to keep our costs as low as possible so we can afford to make it to phase 2. To that end, we must implement caching and cache clearing, low cost threads, and offloading as much processing to the Front End as possible.

* Documentation

After this initial roll out, we will be in Phase 1 of our Business Plan. During that Phase, we will continue to add features as we build a User Base. Effective code documentation is necessary to enable easily adding new features to Comment Anywhere.

* Password security

We need to keep the passwords of our user's secure. We can do this by encrypting the users passwords before storage. Passwords will only be stored in their encrypted state and we will not have access to them.

* Username Policy

We need to have a username policy that describes which types of Usernames are prohibited.

* Comment Policy

We need a Comment policy which describes which sorts of comments are prohibited or otherwise regulated. Our policy must be strict enough to comply with U.S. Law, but not so strict as to alienate our target user base, who prefer freer speech when possible.

* Back Ups

Our system must be prepared for failures. We will need to create regular backups of our database so we can restore from past points without losing too much data when the system goes down.

* General security

We will need to take some steps to make the application secure. Generally, most security improvements will be saved until after the initial roll out but some things, such as basic SQL injection prevention and password encyption should be implemented by the initial roll out.

* Appealing User Interface

Our user interface will need to be user friendly or else users will just not use the product. We will attempt to create a pleasing to the eye graphical user interface (gui).

* Deployability

Our build and deployment setup needs to be flexible enough to change hosting providers as our needs evolve. We can set up our build chain to use Containers, such as through Docker, allowing us to deploy our Product on a variety of cloud platforms, or through self-hosting.

* Legality

Our Program will have to be compliant with United States law. It must have a Privacy Policy and a Terms and Conditions that all Users agree to, to help mitigate lawsuits and other possible legal actions.

Documentation

Many types of documentation will be created throughout the project's lifecycle in order to produce a highly maintenable and extensible product that can adapt and scale after release. The target of the documentation is current and future developers. End users will require little documentation to interact with the product. Documentation for the developers will consist documentation generated from code comments and some additional hand written files.

The standard engineering documentats that will be created are:

1. Proposal Document

The initial document describing the intent of the project.

1. Requirements Document

This document; used to describe the basic needs of development.

1. Specifications Document

....

1. Design Document

....

Additionally, we will create some extra documentation to enable continued Phase 1 development.

1. Testing.md

Will describe a standardized way to write and categorize unit tests and provide instructions for running and writing them.

1. Versioning.md

Will describe the use of git, github and any commit message policies we may implement.

1. Documenting.md

Describes code-commenting practices. It will describe the procedure for generating project documentation from comments and adding additional non-generated documentation to that.

1. Glossary.md

A list of all nonstandard words used. It will be divided into application domain terms and terms specific to this project. It will also have a section describing the contents of each source code folder and a section describing the purpose of each file in root.

1. Build.md

Will provide instructions for building and running the project.

1. Database.md

Will describe database migration and backup procedures.

Testing Revisions

This document underwent several phases of testing. The first phase was unit testing during development, when the document was broken into portions for each section. Each unit consisted of a section's content, associated terms, and references. Every team member reviewed each section in isolation many times and improved on the content, references and glossary before the document was combined into an initial rough draft.

The rough draft was continuously integration tested by running a merge script which assembled a markdown file by combining the text, term, and reference components. The document was evaluated as a whole and improvements were made on component sections to optimize their performance in the rough draft.

After a satisfactory rough draft was produced, the markdown document was copied into Word for quality assurance. Every team member participated in spellchecking, formatting, and adding some additional content in the final word document.

Finally, we simulated an acceptance test by taking the document to the Writing Center for review. We made some minor revisions in response to test results.

Appendix

Technical Glossary

**Administrator**

The highest authority of moderation can do anything a global moderator can do and see their approval to global moderator

**Application Domain**

The specific environment in which the product is to operate. [7] Can be an organization, a department within an organization, or a single workspace. [8]

**Back End**

A Back End is any part of a website or software program the users do not see. It contrasts with the Front End, which refers to a program or website's user interface.

**Bug**

A mistake within a computer program that causes unexpected results.

**Cloud**

"The cloud" refers to servers that are accessed over the Internet, and the software and databases that run on those servers. Cloud servers are located in data centers all over the world. By using cloud computing, users and companies do not have to manage physical servers themselves or run software applications on their own machines. [13]

**Comment**

A line of text created by any user can be replied to by other comments ,edited for a spacific duration reported and become a removed comment

**Database**

A database is an organized collection of structured information, or data, typically stored electronically in a computer system. A database is usually controlled by a database management system (DBMS). Together, the data and the DBMS, along with the applications that are associated with them, are referred to as a database system, often shortened to just database.Data within the most common types of databases in operation today is typically modeled in rows and columns in a series of tables to make processing and data querying efficient. The data can then be easily accessed, managed, modified, updated, controlled, and organized. Most databases use structured query language (SQL) for writing and querying data. [12]

**Domain**

A domain is the name of a website.

**Domain Moderator**

A domain moderator manages all comments under their assigned domain.

**Front End**

The Front End of a software program is everything with which the user interacts. From a user standpoint, Front End is synonymous with the User Interface. The Front End of Comment Anywhere is the Browser Extension. [15]

**Git**

Git is free and open source software for distributed version control. [4]

**GitHub**

An internet hosting service for software development and version control using Git. [5]

**Global Moderator**

Higher authority than a moderator can ban people from any domain or all domains can also see when a regular moderator was premoted and by who

**Guest**

A guest can browse content without a registered account. Guests cannot create content, nor interact with members.

**Inheritance Diagram**

A diagram that relates and identifies similar properties among multiple objects.

**Logs**

Logs are used by administrators to view the technical details of server events.

**Member**

Members can create and interact with public content while using a registered account.

**Nonfunctional Requirement**

Properties of the product such as platform constraints, response times, or reliability. [16]

**Requirements Engineering**

The process of defining, documenting, and maintaining requirements in the engineering design process. [1]

**Server**

A program which waits for a request then performs some service for the requester and which runs on a computer other than the one on which the requestor/client runs. [11] The Server used by Comment anywhere is an HTTP server because it processes HTTP requests on the Internet.

**User**

Anyone that has the browser extention installed on a browser, low level access to the program only able to comment, reply to comments and be premoted to a moderator global moderator or admin or banned by moderators and administrators either entirely or on a spacific domain.

**User Account**

User accounts are used to uniquely identify someone using a particular service.

**User Interface**

Also called a "UI" or "interface", a User Interface is the means in which a person controls a software application or hardware device. [14]

**User Privileges**

A status that indicates how much a user can interact with a service.

**Version control**

Systems responsible for managing changes to source code and other collections of information. [6]

**Virtual Machine**

Also called a Container, a Virtual Machine is a compute resource that uses software instead of a physical computer to run programs and deploy apps. One or more virtual "guest" machines run on a physical "host" machine. Each virtual machine runs its own operating system and functions separately from the other VMs, even when they are all running on the same host. [9]

Team Details

This document was created by the team. We met several times in the Student Center to consult with each other as we worked. We utilized Discord to communicate and GitHub to manage project iterations.

Specific contributions are as follows.

Karl Miller created the project GitHub repository and wrote a python merge script to merge content and references from subfolders into a single large draft. He wrote the initial Abstract, Objective, Team Details & Dynamics, Operating Environment, Documentation, Testing Revisions, and Functional Requirements sections. He edited the Description of Data Sources and Nonfunctional Requirements.

Luke Bates designed and described the use case diagrams. He also added necessary terms to the appendix.

Frank Bedekovich....

Robert Krency ....

Workflow Authentication

Report from Writing Center

References

[1] - <https://www.javatpoint.com/software-engineering-requirement-engineering>

[2] - <https://www.techdirt.com/2021/11/08/killing-website-comment-sections-wasnt-brilliant-move-many-newsroom-leaders-assumed/>

[3] - <https://www.poynter.org/ethics-trust/2021/dont-read-the-comments-for-news-sites-it-might-be-worth-the-effort/>

[4] - <https://en.wikipedia.org/wiki/Git>

[5] - <https://en.wikipedia.org/wiki/GitHub>

[6] - <https://en.wikipedia.org/wiki/Version_control>

[7] - Schach, Stephen R., Object-Oriented and Classical Software Engineering, 8th edition.

[8] - Züllighoven, Heinz, Object-Oriented Construction Handbook, Chapter 6.3, 2005. <https://www.sciencedirect.com/topics/computer-science/application-domain>

[9] - <https://www.vmware.com/topics/glossary/content/virtual-machine.html>

[10] - <https://techterms.com/definition/backend>

[11] - <http://www.catb.org/jargon/html/S/server.html>

[12] - <https://www.oracle.com/database/what-is-database/>

[13] - <https://www.cloudflare.com/learning/cloud/what-is-the-cloud/>

[14] - <https://techterms.com/definition/frontend>

[15] - <https://techterms.com/definition/user_interface>

[16] - Schach, Stephen R., Object-Oriented and Classical Software Engineering, 8th edition.