ANOTODE

The Web Annotator .

SRS Revision 3.0

CS Group 1

Authors:
Avi (201451070)
Saurabh (201451017)

Reviewed by:
Manohar (201451024)
Mukesh (201451060)

Revision Table

Revision	Author	Reviewer	Revision Date	Revision Tracking Notes
1	Saurabh	Manohar	26-9-16	Added intended audience
2	Avi	Manohar	27-9-16	Added requirements and other software attributes
3	Avi	Mukesh	28-9-16	Added use case diagram

Contents

1 Introduction **4**
- 1.1 Purpose . . . 4
- 1.2 Document Convention . . . 4
- 1.3 Intended Audience and Reading Suggestions . . . 4
- 1.4 Product Scope . . . 4
- 1.5 References . . . 5

2 Overall Description **5**
- 2.1 Product Perspective . . . 5
- 2.2 Product Functions . . . 5
- 2.3 User classes and characteristics . . . 5
- 2.4 Operating Environment . . . 5
- 2.5 Design and Implementation Constraints . . . 6
- 2.6 User Documentation . . . 6
- 2.7 Assumptions and Dependencies . . . 6

3 External Interface Requirements **6**
- 3.1 User Interfaces . . . 6
- 3.2 Hardware Interfaces . . . 7
- 3.3 Software Interfaces . . . 7
- 3.4 Communications Interfaces . . . 7

4 System Features **7**
- 4.1 User Account . . . 8
  - 4.1.1 Description . . . 8
  - 4.1.2 Functional Requirements . . . 8
- 4.2 Highlighting Text and saving it on server . . . 8
  - 4.2.1 Description . . . 8
  - 4.2.2 Functional Requirements . . . 8
- 4.3 Searching a user's highlighted notes . . . 9
  - 4.3.1 Description . . . 9
  - 4.3.2 Functional Requirements . . . 9
- 4.4 Adding tags/labels to annotations . . . 9
  - 4.4.1 Description . . . 9
  - 4.4.2 Functional Requirements . . . 9
- 4.5 Re-highlighting text when a user re-visits the page . . . 9
  - 4.5.1 Description . . . 9
  - 4.5.2 Functional Requirements . . . 10
- 4.6 Export user data . . . 10
  - 4.6.1 Description . . . 10
  - 4.6.2 Functional Requirements . . . 10

5 Other Nonfunctional Requirements **10**
5.1 Performance Requirements 10
5.2 Safety Requirements 11
5.3 Security Requirements 11
5.4 Software Quality Attributes 11
5.4.1 Availability 11
5.4.2 Flexibility 12
5.4.3 Testability 12
5.4.4 Reliability 12
5.4.5 Portability 12

6 Other Requirements **12**
6.1 Legal Requirements 12

7 Appendix A: Glossary **13**

8 Appendix B: Analysis Models **13**

1 Introduction

1.1 Purpose

Anotode is a web-integrated note taking application. The motive of taking up this project is to enhance the user experience while web surfing by having a prominent feature of annotation. With this, a user can store everything that he likes on the web in a safe and secure place with minimum hassle and later on find what he needs using the powerful search. With the Android app, they will always have the notes they have taken, right in their pocket. We hope Anotode will help users have a better note taking experience when browsing the web.

1.2 Document Convention

This document uses the same standards as our all other documents and the specifications for the same has been defined in the Document Convention document. Apart from those defined there, in this document we are using some conventions which are as follows -

- The term highlight and annotate has been used inter-changeably in this document. They both mean the same thing i.e. the act of highlighting/underlining/noting a particular piece of text. Annotation means the object that is highlighted text (Noun). Similary the term highlight has been used as a Noun at some places as well.
- By the term client apps, we mean both Web and Android app. In our case, we generally have the same feature set in both these apps so the term client apps has been used extensively for them.

1.3 Intended Audience and Reading Suggestions

This document is intended for developers, testers, users and documentation writers. The Software Requirement Specification is organized into following as Introduction of Project, Overall description, System Specific Requirements and other non-functional requirements.

The reader must go through System Specific Requirements and other non-functional requirements for detailed study of features proposed to be incorporated in project Anotode.

1.4 Product Scope

The project solely concerns with enhancing user experience with web by providing a feature of annotation. The target audience considered in this project are students, research-oriented people, bloggers, developers and all those who use the web for learning things.

1.5 References

The document in this file is an annotated outline for specifying software requirements, adapted from the IEEE Guide to Software Requirements Specifications (Std 830-1993).

2 Overall Description

2.1 Product Perspective

The system proposed in this document provides support for major popular web browsers like Google Chrome, Mozilla Firefox and if possible Safari. The system will function as Web App and Android App to make note taking from external sources efficient and flexible. The system is motivated from Evernote's capture your mind software solution for note taking. The system proposed will focus primarily on highlighting text from the Internet and keeping it integrated with the Internet which is not present in the former.

2.2 Product Functions

The system will provide the following features :

- Chrome and Firefox extension to help to highlight and make comment on text.
- Tags and categories with color feature will help to differentiate highlights.
- Search for tags and categories feature in web app and android app.
- Create a note on demand.
- Export user data as plain text.

2.3 User classes and characteristics

The user base for the proposed system will be students, research-oriented people, bloggers, developers and those who wants to use a flexible note taking application. The assumption made here is that users read text from Internet and appreciate note taking feature. The user must have moderate knowledge of using Internet and is familiar with using Web Applications.

2.4 Operating Environment

The system will be functional as Web App and Android App. The system is compatible with major web browsers like Chrome, Firefox and Safari. The android application will be operable for android Kitkat (4.4.2) and all future versions.

2.5 Design and Implementation Constraints

Below is the list of everything that can put limits/constraints on the choices available to the development team while they build the product.

- We will be using NodeJS for the back-end so we will be forced to use a Linux system for our server. Windows hosts are known to have performance issues with NodeJS.
- Since the user will be taking private notes and then won't want anyone to have a look at what they have recorded, the development team will be forced to use end-to-end encryption while storing notes so that even the system administrators don't have access to user information.
- There is a requirement which involves that highlighted pages should be re-highlighted when a user revisits them. Sending a server request everytime user visits a page is not a good idea. So in order to conserve software efficiency, the development team will be forced to work on some sort of temporary storage in the browser extension so that the highlight data about a page can be retrieved without sending a request to the server. There can be data sync issues too and development/design team will have to carefully take care of it.

2.6 User Documentation

A short video tutorial will be provided for getting to know the functionality of the system. Apart from this, we also plan to make the user manual available using ReadTheDocs, a VCS enabled documentation generation system. It will allow us to generate documentations in multiple formats on the fly; the formats themselves being PDF, EPUB, Kindle (MOBI), web-based etc.

2.7 Assumptions and Dependencies

The assumption made here is that users read text from Internet and appreciate note taking feature. The user have moderate knowledge of using Internet and is familiar with using Web Applications.

3 External Interface Requirements

3.1 User Interfaces

- For the web app and the Android app, we will be following material design conventions and the design will be uniform in both client apps.
- All error messages will include a distinct error code that will help us identify the exact cause of the error.

- For the browser extensions, Ctrl-Alt-H will be the shortcut that can be used by a user to highlight/annotate the selected text.
- Android app will be optimized for Potrait screen layout.

3.2 Hardware Interfaces

- The Android app will be optimized to work on all screens more that 4.0 inch in size.
- The Web app will be mobile optimized to work on at max at 360p width. Less than that will not be supported by our team.
- Annotating a note by selection on touch-screen based systems will be supported by the application.

3.3 Software Interfaces

- The Android app will be linked with the in-built Android Share feature that will allow user to create a note from text in any other app. The selected text from another app will come into the system as input and the output will be a Toast message confirming that the text has been saved into Anotode.
- The server will be using MongoDB database as the database architecture. We are using MongoDB because it is No-SQL, fast and scalable and very well integrated with NodeJS.
- The browser extension will be using the context menu to give the feature of highlighting a text. By clicking on highlight in the browser's context menu, selected text will be sent into our system as input and it will be saved in user's account.

3.4 Communications Interfaces

- Once a user account has been created on the system, we will need to verify it's identity be sending an automated confirmation email to the user's email address. We plan to use a service such as Sendgrid for sending mails. The mails will be sent following SMTP standard.
- To preserve data integrity and to prevent it from tampering, the server will run with HTTPS enabled. All data sent from client apps or browser extension to the server will be encrypted so that data can't be breached in between by any unauthorized third party.

4 System Features

The following is the list of system features that were extracted from the requirements. They have been listed from highest to lowest priority.

4.1 User Account

4.1.1 Description

The system will allow a user to create an account in order to start using it. Once they have an account and are authenticated, they can visit a webpage and start annotating what they want.

4.1.2 Functional Requirements

Following is the list of functional requirements which gave this feature.

- **SR 1** User should be able to highlight the text they read on the browser.
- **SR 4** There should be a website which lists authenticated user's highlights.
- **SR 5** There should be an Android application which lists authenticated user's highlights.
- **SR 7** Everything highlighted on the browser should be viewable via Android and Web app.
- **SR 8** If a user uses multiple browsers, they should be in sync. That is, data highlighted on one browser = data highlighted on another.
- **SR 9** A user account should be secured by password.

4.2 Highlighting Text and saving it on server

4.2.1 Description

A user can visit a webpage and highlight it. Each of their annotations will be saved on our server so that it can retrieved later when needed.

4.2.2 Functional Requirements

Following is the list of functional requirements which gave this feature.

- **SR 1** User should be able to highlight the text they read on the browser.
- **SR 3** A highlight can have a separate color for highlighting.
- **SR 7** Everything highlighted on the browser should be viewable via Android and Web app.
- **SR 8** If a user uses multiple browsers, they should be in sync. That is, data highlighted on one browser = data highlighted on another.
- **SR 12** User should be able to create a custom note (i.e. not by highlighting). Those notes will have no URL.
- **SR 13** User should be able to add a title to a highlight. This feature will be optional.

4.3 Searching a user's highlighted notes

4.3.1 Description

Users will be able to find the highlights/notes/annotations they have done using their account. This search will work across highlighted text, tags and categories. The search will also filter through title and comments to be as powerful a tool as possible.

4.3.2 Functional Requirements

Following is the list of functional requirements which gave this feature.

- **SR 10** User should be able to filter highlights by tags and/or categories from the client apps.
- **SR 11** User should be able to add optional comments to a highlight.
- **SR 13** User should be able to add a title to a highlight. This feature will be optional.

4.4 Adding tags/labels to annotations

4.4.1 Description

The user will be able to add tags, categories, colors etc to a highlight so that they can group annotations together. The search will work across comments and tags to filter specific highlights later.

4.4.2 Functional Requirements

Following is the list of functional requirements which gave this feature.

- **SR 2** They can add tags and categories to what they highlight.
- **SR 3** A highlight can have a separate color for highlighting.
- **SR 10** User should be able to filter highlights by tags and/or categories from the client apps.

4.5 Re-highlighting text when a user re-visits the page

4.5.1 Description

As the highlights are being saved on the server, so a page will have to be re-highlighted when the user visits that page again. This is the essence of note-taking and is the core idea of our application.

4.5.2 Functional Requirements

Following is the list of functional requirements which gave this feature.

- **SR 1** User should be able to highlight the text they read on the browser.
- **SR 6** If a user comes back to an article, the browser should show what they have highlighted.
- **SR 14** When using Google, a user's highlighted articles should stand out from other search results.

4.6 Export user data

4.6.1 Description

The user should be able to export their data and save a copy offline. This will give a sense of security to the user whil using our application.

4.6.2 Functional Requirements

Following is the list of functional requirements which gave this feature.

- **SR 15** User should be able to export their data in plain text format.

5 Other Nonfunctional Requirements

5.1 Performance Requirements

- The browser extension should be quick and efficient to display the previous highlights on a webpage. For that, the development team will be forced to use some form of data caching in the browser extension so that server is not required everytime a new page is visited.
- The search feature should not take too long to be executed. Since we plan to develop a more human friendly space based multi-search, we need to make sure that this doesn't make the search feature slow.
- If the number of highlights done by a user is very high (suppose ¿ 1k), then in that case returning all data at once to the client apps won't be feasible. The client apps will then have to implement some sort of lazy loading so that data from the server is loaded in parts. This will also help reduce load on the server.
- Search feature can place huge load on the database and the developer team may be forced to use data sharding, database duplication or other techniques to ensure fast database access.

- The website (web app) needs to be fast to load so that user experience is not affected. For that, we will implement gzip compression on server to reduce the size of assets and use public CDNs for libraries to reduce framework overhead of a webpage.

5.2 Safety Requirements

- Since user data will be of utmost importance in our app, once the project gets popular and widely used, we will starting using database backup services to keep our data across multiple data centers. This will give both us and the users that their data will be safe on our service.

5.3 Security Requirements

- The main concern for a data-based project is indeed hacking and data breach. To ensure that user data can only be viewed by the user themselves, we will use end-to-end encryption over data. In this, we will store the user data in encrypted form in our database and only through user password, a person will be able to decipher that data.

- A lot of users use common passwords these days for multiple accounts, so to prevent Anotode admin team from accidentally viewing a user's password, we will store hashed form of the password in our database rather than the password itself.

- We will use JWT (JSON web tokens) for authentication on our server. JWT is a database-less standalone authentication system with extended features such as expiry time of a token. If we use it, a hacker will never be able to take control of a user's account completely unless he knows the user password.

- To be even more sure about security, we will get our system tested with a team of Ethical hackers before the public release (i.e. in beta).

5.4 Software Quality Attributes

5.4.1 Availability

- Ours is an web based application so generally it will only work if one is connected to the Internet. This will work for Browser extension and Website but for Android app, user may expect to be able to view their highlights even without a network connection. So we will implement caching/offline store in Android app so that data that is once fetched from the server is saved on the phone so that it can be viewed later on. Also that data can serve as a placeholder when the latest data is being fetched from the server, so it will also improve user experience.

5.4.2 Flexibility

- We plan to make the system flexible for the user by not enforcing any tags/categories that they can use. They can write any word and it will become a category/tag and will be search-able.
- The user will not be limited to a particular type of text that they can annotate. They can highlight anything from article title to a youtube comment to a image caption and save it using our service.

5.4.3 Testability

- We plan to heavily test the server and the client apps using continuous unit testing feature of Travis CI. We will try to maintain a high coverage for the projects and make sure that every line of code is tested.
- As we are using Incremental SDLC model in our project, we will be doing beta releases and inviting testers to test the application. This will help us catch lots of bugs.

5.4.4 Reliability

- The server will be hosted on a premium hosting provider (Digital Ocean, AWS) to ensure high availability.
- The client apps, the server and the browser extension will undergo beta testing by volunteers to ensure that the bugs are caught before making the project public.

5.4.5 Portability

- Since we are only making Android app right now, to be able to use Anotode on other mobile platforms like Windows, iOS, we will make the application's website mobile-responsible by sticking to a popular web framework like Bootstrap.
- User can take out their data any time from our system by using the export feature.

6 Other Requirements

6.1 Legal Requirements

- We will be storing data of different websites on our server. This can lead to copyright infringement in some cases but since we are storing source url with every piece of data stored and the scope of data is strictly limited to personal use, there should be no legal issues with our application. Similar applications like Evernote, Pocket are active proofs for this.

7 Appendix A: Glossary

Annotation (Noun) Same as Highlight (Noun)

Anotode Our product name

Extension Extension refers to Browser extensions that are being developed in this project. These include Chrome, Firefox and possibly Safari extension.

Highlight (Noun) A piece of text that has been annotated by the user

Highlight (Verb) The act of highlighting or annotating

HTTPS Hyper Text Transfer Protocol Secure

JWT Json Web Tokens. A modern stateless O-Auth mechanism. See jwt.io for more info

ReadTheDocs An online site to host documentations, supports VCS.

Server Server refers to the backend server which is responsible for serving as a medium so that the client apps can access the database.

Travis CI An online VCS-integrated unit test runner. For more information, see travisci.com

VCS Version Control System. Used to keep track of changes in a document

Web App Web Application, a website with interactive features

8 Appendix B: Analysis Models

The following is the use case diagram for this project. It identifies how different types of users interact with the system.

case diagram.png