Software Requirements and Design Document

For

Group 10

Version 1.0

Authors:
Aidan Martin
Riley Corey
Darren Kopacz
Douglas Kendall
James Kerrigan

1. Overview (5 points)

WebSight is a web-application that stores a list of website URLs provided by a user, and sends an alert to that user whenever a website is updated. The site runs from a main page, which offers a dynamic table of URLs that the user has decided to track. From that main page, users can see the date added, and expiration date of any given URL as well as manage alerts to set timeframes to check for updates and receive alerts. When adding a new website to track, users are solicited for a valid URL, their preferred method of communication, tracking duration, and whether they are looking to track specific content, such as words, images, or any change. Users also have the ability to remove, or stop receiving alerts from any given site. The site is self explanatory and easy to use/navigate without any instruction.

2. Functional Requirements (10 points)

LOG-IN PAGE:

1. *HIGH - Ability to create an account or log-in if an account already exists. Else display error messages. [Accounts necessary so that a user's chosen websites can persist indefinitely if they so choose].*
2. *MID - Textbody describing what WebSight is and how it works.*
3. *MID - Enforce password length/character requirements.*
4. *LOW - Forgot password/reset password feature.*
5. *LOW - Bot prevention*

MAIN PAGE:

6. *HIGH - Ability to add a URL, its nickname, and alert option data to a user-specific tracking list (must persist between log-in/log outs).*
7. *HIGH - Log-out button brings user to log-in page.*
8. *HIGH - Check if that URL is valid.*
9. *HIGH - Manage Alerts button links to Manage Alerts Page*
10. *MID - Display X number of URL nicknames in Current Alerts Dropdown (toggles).*
11. *MID - See More button at bottom of Current Alerts. Link to Manage Alerts page.*
12. *MID - Adding a website/Alert instantly updates list or refreshes page to update it.*
13. *LOW - Option to remove a website in the dropdown menu.*
14. *LOW - Each URL listed in dropdown menu links to a sub-menu in Manage Alerts.*
15. *LOW - Ability to switch accounts instead of just logging-out. Dropdown options?*
16. *LOW - Visual Comparisons*
17. *LOW - Ability to display Current Websites dropdown in order of most recently updated, most recently added, etc.*

MAKE CHANGES/MANAGE ALERTS PAGE:

1. *HIGH - Ability to modify any of the options chosen during the add alert menu.*
2. *MID - If no websites added, present option to add a website (links to main page).*
3. *MID - Link back to Main Page.*

4. *LOW - Separate list of “expired” website tracking.*
5. *LOW - Option to re-add an expired website.*
6. *LOW - Change username, password, email, phone.*
7. *LOW - Option to remove ALL saved websites at once.*

3. Non-Functional Requirements (10 points)

Security

1. *Usernames will be unique and require a specific amount of characters.*
2. *All username/passwords will be stored in the database. Eventually, password’s will need to be hashed to increase account security.*
3. *User is able to log in with a username and matching password.*
4. *Passwords should display stars as typing.*

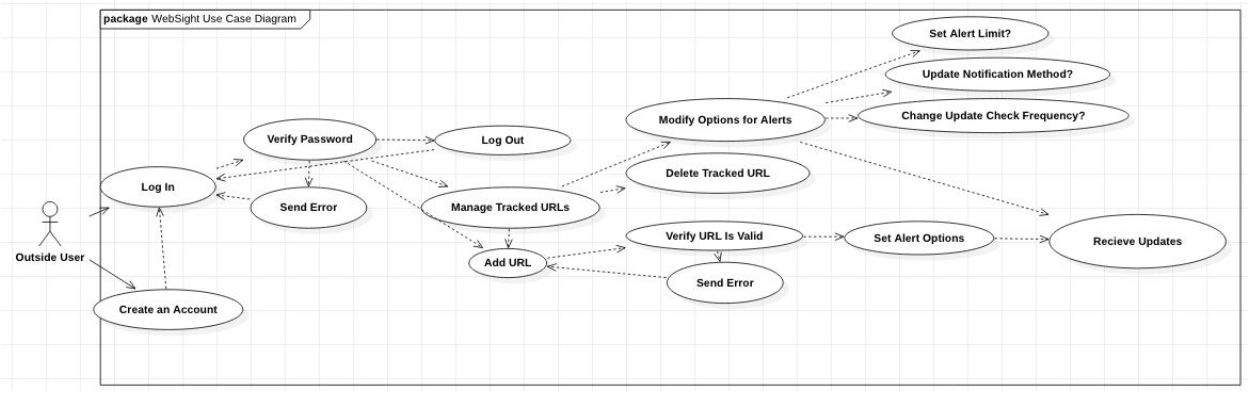
Safety

5. *User data involving website URL and tracking options must be saved to each account uniquely (‘User’ object with urlList object inside).*
6. *Users cannot create an account with an existing username (username list?).*
7. *Users must verify their account by email. This confirms a user will receive email-alerts, and helps prevent misuse of the website update-alert system for spamming or other malicious intent.*
8. *URLs not accessible from the server will return an error*

Performance

9. *Current Websites dropdown should be displayed in order of most recently added.*
10. *Dropdown list implies the website should only display X number of characters before being cut off and followed by ‘…’. If user chose to give the website a nickname, display that instead, with an internal cap on the # of nickname chars.*
11. *Removing a website from Manage Alerts page must remove it from the user’s unique urlList.*
12. *Limit emailing/alerting users when a selected site changes constantly.*
13. *OPTIONAL: Any url added by any user will be saved to a master list. This way, commonly checked websites only need to be checked once per shortest-time interval, rather than having the server check the same website for 100+ times within every minute. However, any time a user deletes any website, the masterList must be checked to see how many people are still tracking it. If less than X users are tracking that site, remove it from masterList and let user’s clock-time decide when to check (unless there would be a syncing issue?).*
    REQUIRES SYNCING TO SERVER TIME (basically, if user chooses 5 min intervals at 12:03, it will start checking at 12:05).

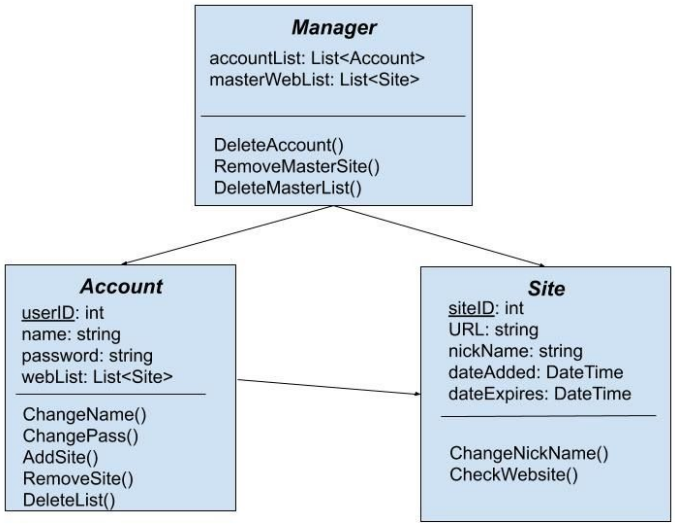
4. Use Case Diagram (10 points)



The homepage offers two options: sign in, or register an account. If the password is verified, the next page presented is for managing tracked URLs. If not, an error is presented on the log-in/home page. Once logged in, the user can add or delete tracked URLs, and modify alert options for previously added URLs.

Adding a URL to the tracked list requires verification of its validity. If valid, the user is presented with alert options, if not, an error is presented.

5. Class Diagram and/or Sequence Diagrams (15 points)



A single manager class instance handles all of the accounts and sites on the application. It interacts with the Account class and Site class to hold lists of Account objects and Site objects. The Account class allows users to create an account that includes their username, email, and password, and is linked to a list of Site objects. This class allows modification of username, password, and site list. The Account class interacts with the Site class, which creates Site objects for each URL added. It includes the website's URL, a nickname, date added, and an expiration date.

6. Operating Environment (5 points)

WebSight (V2) is a web application that has been developed on the ASP.NET framework within Visual Studio. It is hosted through IIS Express. It has been created to properly function on any major browser, including Safari, Google Chrome, Firefox, and more.

7. Assumptions and Dependencies (5 points)

Dependencies currently include the SendGrid API for SMS text alerts and email, and a to-be-determined API for saving website HTML data for analysis. However, API was never integrated.

WebSight V2 is dependent on a local connection string to a local database. The connection string will be different for each computer therefore it will need to be changed.

1. Programming Languages (5 points)

We have chosen to use Hypertext Markup Language (HTML) and Cascading Style Sheets (CSS) for our web application interface. We have chosen these due to the fact that HTML is a standard to use for designing web pages and CSS is used for advanced styling. We have also chosen to compose our class functions with C# provided that all team members have been well introduced to the C++, C or Java.

WebSight version 3 includes much more Javascript, including HTTPs requests that communicate with our localDB. By using Angular, data is transmitted and displayed onto the applications front-end.

2. Platforms, APIs, Databases, and other technologies used (5 points)

In the last increment, we shifted our application to the ASP.NET Core framework to allow development on Apple machines, but the shift caused many problems, forcing us to revert back to the ASP.NET framework in Visual Studio. We continued to use Microsoft SQL to handle any databases, but decided against using the SendGrid API and Azure.

In this increment, we shifted back to ASP.NET and used SQL for our database and data tables. We used the IIS Express Web Server to host our site locally.

3. Execution-based Functional Testing (10 points)

Local communication between application and DB is established using GET (HTTPs) methods, but due time there were difficulties in presenting working POST methods, so appending data is not functional.

4. Execution-based Non-Functional Testing (10 points)

We have tested that the application runs successfully on various machines (Windows/Mac) and various web browsers (Safari, Chrome, Firefox, etc). We have tested that the site runs efficiently, without any

pages or additions/access to databases taking an extraneous amount of time. We will soon test site security to ensure that user's data is securely stored and hackers are not able to compromise the site by entering data into any request fields. We have tested the site for usability to ensure that users are able to operate the site without requiring any additional instructions or training. While the site works functionally on various browsers, it currently only runs on Windows machines and causes issues on Macs.

5. Non-Execution-based Testing (10 points)

Overall, the back-end logic looks fine. The front-end logic needs quite a makeover as it's a blob of code that's structured through inexperience. Separation of the newly implemented Javascript could have been separated from the HTML/CSS page if not due to time constraints.