• Progress Report

- Increment 1 -
Group #10

1) Team Members

Name	*FSUID*	*Github ID*
Aidan Martin	*am16br@my.fsu.edu*	*am16br*
Darren Kopacz	*dk16d@my.fsu.edu*	*dk16d*
Riley Corey	*rjc18d@my.fsu.edu*	*rileyycoreyy*
Douglas Kendall	*djk12@my.fsu.edu*	*dougkendall*
James Kerrigan	*jk18ed@my.fsu.edu*	*KerriganJames89*

2) Project Title and Description

WebSight takes a website's URL and solicits contact information from the user, such as an email or phone number, and notifies the user any time the website is edited or updated. We have a log in/sign up page for users to create an account before they are able to add to a list of website URLs. Our framework connects to a database that holds user information and a list of URLs, which the code checks intermittently for changes to the site's source code. Once the website is updated, the user receives a notification via their preferred method. Our application functions as a website that will be easily accessible from any major web browser. Users are able to edit their list of websites, choose their preferred method of communication (email or text), how often they would like to check for and receive update information, and if they are interested in checking for a certain change.

3) Accomplishments and overall project status during this increment

In this increment, we created a graphic user interface that drives the application's functionality. A framework was created that will interface with a database that stores data in the form of the user's relevant personal information and a list of websites. We have made good progress and moved past our plans for completion in this increment. The framework covers the relevant functionality of checking a site's source code for updates, creation of user accounts, and a connected GUI.

4) Challenges, changes in the plan and scope of the project and things that went wrong during this increment

Due to the short summer semester, there were a few problems when brainstorming for ideas with the lack of time. The project started out focusing only on a web browsing extension, but expanded into a much larger project. There was a greater scope of plans, but it was needed to reach our goal. A significant amount of time was spent trying to build basic web-pages and a user database before a streamlined framework was found that suited our needs.

5) Team Member Contribution for this increment

a) Progress report, *including the sections they wrote or contributed to*

Darren*: Team Members: Created a table for team members to add their contact info.*

Aidan*: Project Title and Description: Added a description of WebSight that focuses on the application's functionality and features. Accomplishments and overall project status during this increment: Outlined accomplishments and completion of different aspects of the project as it currently stands.*

James*: Functional/Non-Functional Requirements: Contributions to conceptual ideas of the project implementation and features. Primarily relating to the database and information gathering.*

Doug*: Functional Requirements: Contributions to notification system requirements. Contributed to script for the video that James made.*

b) Requirements and Design Document (RD)

Riley*: Use Case Diagram: I diagramed our Use Case Diagram in order to organize our system/software requirements for our web application. It was important to create this in order to visualize and organize the implementation parts we need especially when our software is not in development yet. It is also helpful to see how our cases interact with one other in order to begin the integration of their relationships in our development.*

Darren*: Class Diagram: Since C# is a heavily object-oriented language, I created a simple class diagram illustrating the main entities with their properties and methods that will be used in the project. At the moment, there is currently Account, Site, and Manager. Manager will be used to keep track of users and perform administrator levels tasks. More classes will likely need to be added to this diagram in the next increment as we begin working with the web-alert system.*

Functional Requirements: I added a list of functional requirements broken into three main categories: Login Page, Main Page, and Make Changes Page. Each page has the requirements listed in order of high to low priority.

Non-Functional Requirements: Non-functionals were broken into three categories: Security, Safety, and Performance. Since users expect to be able to save and modify their list of tracked websites at any time, WebSight needs to be secure enough to ensure users can only access their own accounts. This also prevents malicious or accidental misuse of the alert feature to spam other people's emails or phone.

Operating Environment; Assumptions & Dependencies: Added info on the current operating environment and planned API dependencies.

James: *ER Diagram: I created a basic three table diagram: Users, Accounts, Changelog. The graph represents a very minimal presentation of how data will be stored in our tables and the relationship between each table. Users will have to create a unique username, and a password. Each user will create their own unique account; a specific ID, and perhaps more personal information can be appended to this table. The Changelog (a master list) will be the archive for URLs, which can only be accessed by the accounts that submitted them. From here, it needs to be decided if compared data should be reinserted back into a table, else all data comparisons will only be done when a user enables the specific query.*

Aidan: *Contributed to conversations regarding non-functional and functional requirements. Added ideas to functionality of the application, including user registration, log-in, and site source code checking.*

Doug: *Slight reorganization, style changes, and added to the plans for the next increment*

c) Implementation and Testing Document (IT)

Riley*: Programming Languages; Platforms, Databases, API, etc; Execution-based Functional Testing; Execution-based Non-Functional Testing;Non-Execution-based Testing: I listed and explained our reasoning for the languages we chose to implement in this project. I also just stated we have not gotten far enough in our implementation process to have testing yet (not required). Since sections 3-5 are not required for this increment I just mentioned we have not had execution-based testing.*

Darren: *Programming Languages; Platforms, Databases, API, etc: I made revisions to these sections after confirming what services and API's we'd be using to host our web application and alerting users.*

Aidan: *Contributed to decisions regarding preferred programming language and implementation. Ran basic Execution-based Functional Testing in the Visual Studio ASP.NET Web Form environment.*

James*: Programming Languages; Platforms, Databases, API, etc; Execution-based Functional Testing; revisions made to user queries and data collection. Even though the framework is fairly new, I dynamically tested what was available: the user account creation.*

Doug: *There was not much to test for this iteration.*

d) Source Code

Darren: *I used Visual Studio to create an ASP.NET web-app framework using Web Forms to simplify the construction of the GUI by drag-and-dropping elements such as buttons and textboxes. Using one of the built in Authentication features, I changed the authentication method to Individual User Accounts which constructed a user-account database for the site. I was originally building the page from scratch, but ultimately decided to stick with the default pages provided by the framework to modify later.*

The ***Site.cs*** *class can be found within the Models folder and includes properties listed in the Class Diagram. There are also constructors that essentially set null values but always sets the date added to the current date time, and the expiration date by default to 24 hours.*

Aidan: *Created the* ***Account.cs*** *and* ***Manager.cs*** *files added into the Models folders. Both files include the properties listed in the Class Diagram. The Account.cs file has two constructors, a default one to set null values and a parameterized constructor to add a user's data. The file has member functions to change the user's name, password, add a site to the list, remove a site from the list, and clear the site list. The Manager.cs file has two constructors, a default one to set null values and a parameterized constructor to add to the lists. The file has member functions to remove a user from the Accounts list, remove a site from the master list, or clear the master list.*

James*: Currently designing the main page. The website adapts a simplistic design, so the user will spend their most time on this page. Need to add functionality with our database to update*

any requested data from the user. This will allow users to append and view their URLs after logging in. After, will add functionality allowing the user to interact more with the data such as pop-out information where users can update their notifications. I'm deciding on implementing jquery, if I can get it to work that is. That would allow me to insert the "DataTables" plugin which will allow more search/filtering utility for the user.

Doug: *Working with html, javascript, and css to polish the webpages. Work during this iteration was primarily with design requirements. In the future I will be designing the login page.*

e) ***Video/Presentation***

Aidan*: Created PowerPoint to support video with relevant descriptions, accomplishments, source code examples, and display of functionality.*
https://docs.google.com/presentation/d/1WnywW-ayvZKsj8b6PmtloWUmHzP4kcpsnnSYVhHKuf8/edit?usp=sharing

James: *Presented the video and uploaded it to youtube. Everyone contributed information for the presentation in the form of a text document.*

6) Plans for the next increment

In the next increment, we plan on developing the functionality of fetching data web and updating over a certain timeframe. We will work on implementing the SendGrid API to notify users about changes on the site and verify their accounts through email.

As with fetching the data from websites, we must also create the database that stores these URLs which can be later compared to each other. This data will have to be private to only that user which requests it. This will be done by creating a database of user accounts in which we can link this data to. Further details of database design will be decided and implemented during the next increment.

Advance functionality such as being able to visually see the specific changes between updates with a more advanced GUI. This seems doable, but will be left till the end.

7) Link to video

https://www.youtube.com/watch?v=xrJU0QodFgw&feature=youtu.be