Progress Report

- Increment 2 -
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

Our application, WebSight, is a web based software that allows users to monitor a list of websites. Our platform will allow users to create an account, add website URLs to a list, receive updates any time a site is updated, and manage alerts to change settings, such as preferred method of notifications, timeframe to check for updates/receive alerts, and changing expiration date for sites.

3) Accomplishments and overall project status during this increment

In this increment, we shifted the application from ASP.NET to ASP.NET Core to function universally. We created the homepage that will display a table with the user's website URLs to check, date added, and expiration date. The homepage also has links to log in/out and manage alerts. The registration/log-in page has been created and is linked to a table that stores user information, such as username, email, and encrypted password. We are working on a table to store and reference URL data to check the site's source code for updates. We are also working on integrating the API SendGrid in order to send email confirmations when registering for a new account and also for sending the alerts for the website updates. DataTables & Jquery implemented to support our user queries.

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

In this increment, we struggled with collaboration and scheduling meetings due to the remote nature of this class with group members not all present on FSU's campus. We also ran into issues running the ASP.NET on Mac devices, since the Mac version of Visual Studio does not support the current

version of ASP.NET. We worked to transfer the application to the universally accepted ASP.NET Core, but have had some issues with recreating the application in the new environment.

The Visual Studio application available for Macs does not offer the same tools and interface. It does not offer the Server Explorer used to create our databases and tables. It also does not let you see the design view of our application due to the fact that the designer is also not supported on the Mac version. So the Mac users on our team were very limited on what they could help with. However, many had other PCs they could use to contribute more after hours of trying to find the Mac ways of doing certain tasks.

This transition has broken some functionality, such as our login page, which will require additional work reconstructing it. We are currently debating ways to merge our previous framework without much loss.

Robust testing on SendGrid is unavailable at the moment, but we worked on sending emails using Sendgrid to better understand future integration

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: Added changes to application and tasks accomplished during this increment. Challenges, changes in the plan and scope of the project and things that went wrong during this increment: Added some basic challenges we faced during Increment 2. Plans for the next increment: Added tasks to be accomplished during the next increment to ensure that the final application is functional.*

James*: Challenges, changes in the plan and scope of the project and things that went wrong during this increment: discussion on the framework switch and how its affected the project. Accomplishments and overall project status during this increment: Added some accomplishments this incrementation. Plans for the next increment: More discussion regarding the framework switch and what needs to be addressed.*

Doug*: Added to challenges regarding sendGrid integration*

Riley*: Accomplishments and overall project status during this increment: Added to status update about the SendGrid API that we are currently working on integrating. Challenges, changes in the plan and scope of the project and things that went wrong during this increment: Also added to our challenges, specifically for Mac users.*

b) Requirements and Design Document (RD)

(Consists of information pertaining to the previous increment; no changes have been made to these initial documentations)

***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. Class Diagram: added textual description to clarify diagram and links between classes.*

***Doug:** Comments for Use Case diagram*

c) Implementation and Testing Document (IT)

***Riley**: Programming Languages; Platforms, Databases, API, etc; I listed and explained our reasoning for the languages we chose to implement in this project. Also explained the platforms, databases, and API we plan to use and what has changed from increment 1 to increment 2.*

***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. Execution-based Functional Testing, Execution-based Non-Functional Testing, Non-Execution-based Testing*

***James**: Programming Languages; Platforms, Databases, API, etc; changes made to the API's being used with our database. Non-Execution-based Testing : discussion made regarding the general direction and structure of each webpage.*

***Doug:** Testing regarding Sendgrid emails and setting up API*

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 (**HomePage.aspx**). The website adapts a simplistic design, so the user will spend their most time on this page. Implemented DataTables API and inserted into the page that allows users to easily navigate through their data. Data that is not displayed is cached onto the user's device, thus allowing a single query to present all of the user's*

related data. With this API in place, I can now implement elements which will allow the user to interact with these tables, and thus keeping the size of the webpage minimal.

***Riley**: I redesigned the homepage in order for it to be the first site run by the solution instead of the default. I also designed the login page (**login.aspx**) (**login.aspx.cs**) based off of the homepage design created by James. I added the username, email, and password fields in order to register or login to the account created by the user. I integrated the GUI with the homepage which allows the user to click the login button and redirects to the login page. It also has a button to go back to the home page for added support. I am currently creating the database that will work behind the login to store the user information. I am also creating the means necessary for registering a new account and gaining an email confirmation through SendGrid. Eventually a user will not be able to access the homepage until they have registered or logged into their account.*

***Doug:** Setting groundwork for email notification via sendgrid. Added code to the "Controllers" folder*

e) ***Video/Presentation***

***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 finalizing the pages and ensuring that they properly link with one another, such that all functionality is achieved without any errors. We will query the site's source code and compare to previous versions to analyze for any changes. We will also work to ensure that users are notified via email after any sites are successfully found to have any updates or changes. Once notifications for site changes are successful, we will essentially be finished with development and only need to iron out any bugs and ensure that all exceptional cases are properly handled.

We need to fix our login page for our final increment. As our current one does not work with Core framework, we will have to find an alternative or build one from scratch. Plans to merge work done on Core back to .Net framework if possible.

7) Link to video

https://youtu.be/ArQIRaNm81w