Final Report
Version 1
5/5/2022
Rehab Remote

Project Sponsor
Dr. Zachary F. Lerner
BiOMOTUM, Inc.

Faculty Mentor
Felicity H. Escarzaga

Team Members
Kylie Cook
Brandon Roberts
Robert Bednarek
Katarina Marsteller

Table of Contents

Introduction

In the United States, 1 in 345 children has cerebral palsy, a set of neuromuscular disorders effective around birth. Cerebral palsy can reduce the mobility of those that have it. Around 58.9% of children with cerebral palsy can walk independently, 7.8% walk with assistance, and 33.3% have limited or no walking ability [1].

The current treatments for cerebral palsy consist of medication, therapy, and surgery. Biomotum stands within the therapy category by developing a battery-powered ankle exoskeleton that increases walking speed by 32%, stride length by 21%, and improves efficiency by 29% [2]. The business currently works with children with cerebral palsy. However, they wish to expand to people with other conditions that reduce their mobility, such as strokes, surgeries, broken bones, and more.

Biomotum's current workflow starts with the use of the ankle exoskeleton. First, they place a child on a treadmill with the exoskeleton and other physical assistance. Next, the researchers measure how much energy the child uses during testing and track the child's movements. The test is then repeated with different forms of physical assistance while collecting data. After a session of the exoskeleton's use, the patient will access their data through the Biomotum phone application. The patient can visualize their steps, tork, duration of use, and more within the application.

The issue within the business workflow is a lack of desktop visualization. Biomotum currently extracts the raw data by unhashing the file and running the data through code on MATLAB. By the end, the researchers visualize the data through MATLAB, a series of steps that the average patient could have difficulty understanding.

Team Rehab Remote has created a solution for Biomotum's problem, by developing a web-based portal that visualizes the data collected from the ankle exoskeleton. Some key functionalities of the web application include: admin ability to assign exos and approve accounts, user ability to view each exoskeleton, user, and trial which they have access to, and ability to download raw CSV data of specific trials. These functions will help Biomotum in their process of data analytics as they continue testing their exoskeleton prototypes and gathering more data.

Process Overview

We used many different tools to help us throughout the development of this project. To start off, we created a team procedure where we laid out specific rules we have about the team and how we communicate with each other. The main communication platform we used was Discord. Everyone on the team was very familiar with Discord and it has been a very effective tool in communicating with each other as well as attending our weekly team meetings. For tracking the progress of our project and ensuring that we know what tasks everyone is working on, we used Trello. Creating a simple board has allowed us to efficiently manage tasks for the entire team. The web portal itself was developed in Wix so all of the code for the project is stored in the cloud through the Wix platform. On top of this, we also saved a copy of the custom Wix code into a git repository so that we have a backup of the code files just in case.

Requirements

The project must have multiple requirements implemented to cover the previously explained solutions to Biomotum's problem. In the following sections, the topics of functional requirements (functions to the application), performance requirements (how the functions will perform), and environmental requirements (constraints to the project) are covered. The web application must cover a secure way for Biomotum researchers to access all of the data collected by the deployed ankle exoskeletons and for the users only to have the ability to access their data. It also needs to have the ability to display the data in ways that the average user can understand.

Functional Requirements

The functional requirements are the required functionalities that need to be present in the Biomotum web application:

- **User Account System**
  To display specific exoskeleton data to distinct users, the web application will need to have a user account system. The system will connect a specific user to the data they are authorized to see. It will also allow for an administrator user to log into the web portal and see the global data of all patients that are a part of the Biomotum program.

  - **Username and Password Database**
    A database will need to be used to store the usernames and passwords for all web portal users. That way, when users log in, the database will be searched through to find their saved information and send them to the page with their specific exoskeleton data..

    - **Data Pinging**
      In order to keep accurate data on the web portal at all times, the Biomotum AWS (Amazon Web Services) Database needs to be pinged every couple of seconds. This way, every CSV file can be implemented into the web portal and the visualizations can always be up to date.

- **Data Visualization**
  One of the main requirements from Biomotum is to visualize the data that they collect from their ankle-exoskeletons so that users can understand how to track their progress. The data is currently in large CSV files that can be difficult to understand by the average person. By taking the data and turning it into color-coded graphs, the patients can better understand the information.
- **Hierarchy of Data**
  A specific request from the client is to include a hierarchy of data that the admins will have access to. The data will be visualized into graphs, as previously explained. The administrator

will log into the administrator account and immediately see the top of the hierarchy data visualization, which is global data, then hospital/physical therapy data, then patient data.

- **Global Data**
  The global data will consist of one graph of visualized data for the collection of hospitals/physical therapy units that use the ankle-exoskeleton. Then, in the global data section, there will be an ability to view the next section in the hierarchy, which is the individual hospital/physical therapy units.

  - **Hospital/Physical Therapy Data**
    The hospital/physical therapy data will consist of multiple graphs that visualize the data for each area that uses the ankle-exoskeleton. Each graph will contain the data for the specified hospital/physical therapy unit that it maps to. Within each hospital/physical therapy section, there will be an ability to view the next section, which is the patient data.

- **Downloadable Data**
  The web portal will include a download ability that allows for the users to download either the graphs (visualized data) displayed on the screen or the raw data in CSV form.

Performance Requirements

The performance requirements describe how the functional requirements are expected to perform:

- **Database Pinging**
  The exoskeleton stores its data in an AWS database. This database will be accessed with consistency to ensure the user's data is up to date. The data, if needed, will be updated on the web portal and visualizations.

- **Data Load Time**
  The data will be expected to load within a reasonably short period of time for both patient data and administration. The graphs and charts will be loaded at an equally fast time frame to ensure the visualization is given to the user.

- **Easy to Use Interface**
  Both administrators and patients expect a clean environment for viewing their data collected from the exoskeleton. Therefore, ensuring that the interface is user friendly with cleanly dictated tabs in a taskbar will avoid navigation issues and confusion on the location of certain features, creating a quick navigational environment.

Environmental Requirements

The following section will cover environmental requirements. The Environmental Requirements will cover the features and requirements that must be worked around or with. There are not very many environmental requirements. These requirements will include external features that the team must conform to, such examples may include the client's current technologies.

- **The Dataset**
  The dataset is a major restriction on the type of information that can be displayed. The data set is limited to the information provided by the device in use, should there be any other data information, it will be limited to the abilities of the device.

- **Web Hosting Technology**
  The web hosting technology that is used (Wix) will be a major environmental challenge/requirement. The rules and restrictions from Wix will be a major determining factor when developing the web portal. Such restrictions could be listed as programming languages that are compatible with Wix (ex. Javascript) which is the only language compatible with the modules that Wix provides.

Architecture and Implementation

Figure 1: System High-Level Architecture

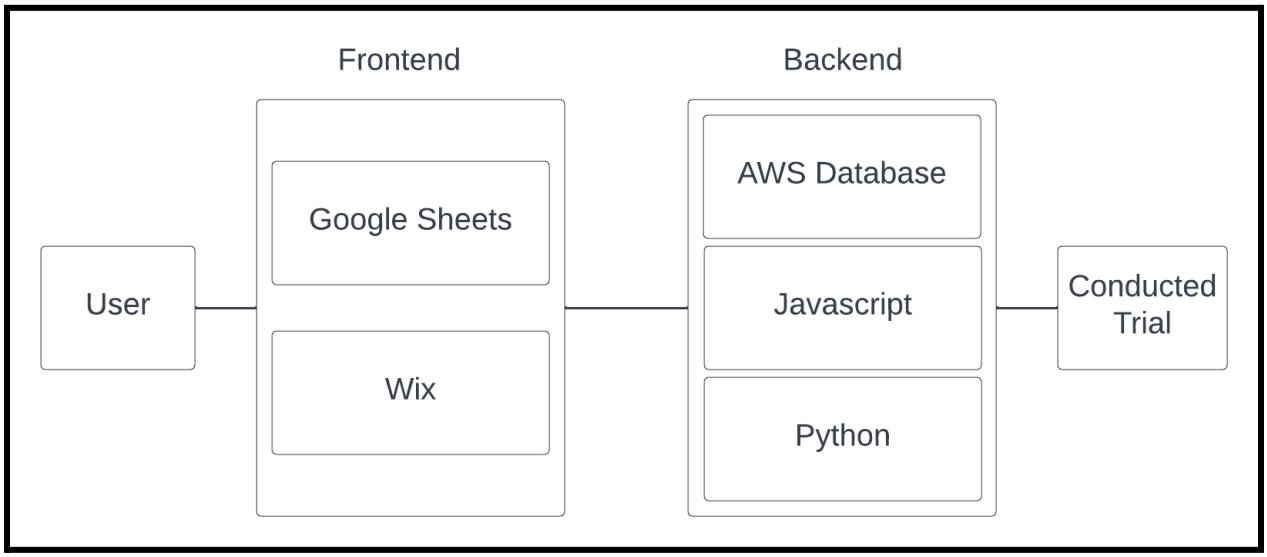


Figure 1 shown above shows the web portal's high-level architecture. This architecture is what forms the website as a whole and makes every section provide necessary data and outputs. In the next few paragraphs, the architecture will be explained in detail, starting with the backend.

The backend of the system contains three main languages and applications, being the AWS (Amazon Web Services) database (S3 bucket), JavaScript, and Python. The AWS database is where all of the CSV (Comma-Separated Values) files created by the Biomotum exokseletons are stored. This is considered the database and backend since the file management system is constantly accessed by different codes throughout the system. The AWS database is only accessed by different parts of the code, it does not access any part of the system itself. Next is the JavaScript, which is used within Wix's backend code. Wix only uses JavaScript for both front and backend. The responsibilities of this part of the system is to collect data from AWS and Google Drive by reaching into AWS, collecting folder names for hierarchy and exoskeleton assignment display, and collecting the data within the summary sheets of the Biomotum Google Drive. Finally, Python is implemented through scripts that are stored in the AWS Biomotum EC2 instance. There are three main scripts written in Python that access AWS S3 to gather data and access Google Drive to create data. The script 'functions.py' is the function script, which holds some main functions used by 'main.py' and 'summaries.py' and is necessary to avoid repetition in code. The 'main.py' script is used to populate the Biomotum Google Drive with the applicable CSV files stored in the AWS S3 bucket, as well as update logs that are also stored within the Drive. The 'summaries.py' script reads through all of the necessary files from AWS S3 and populates the summary Google Sheets stored within the Google Drive with a variety of statistics.

Figure 2: JavaScript Architecture

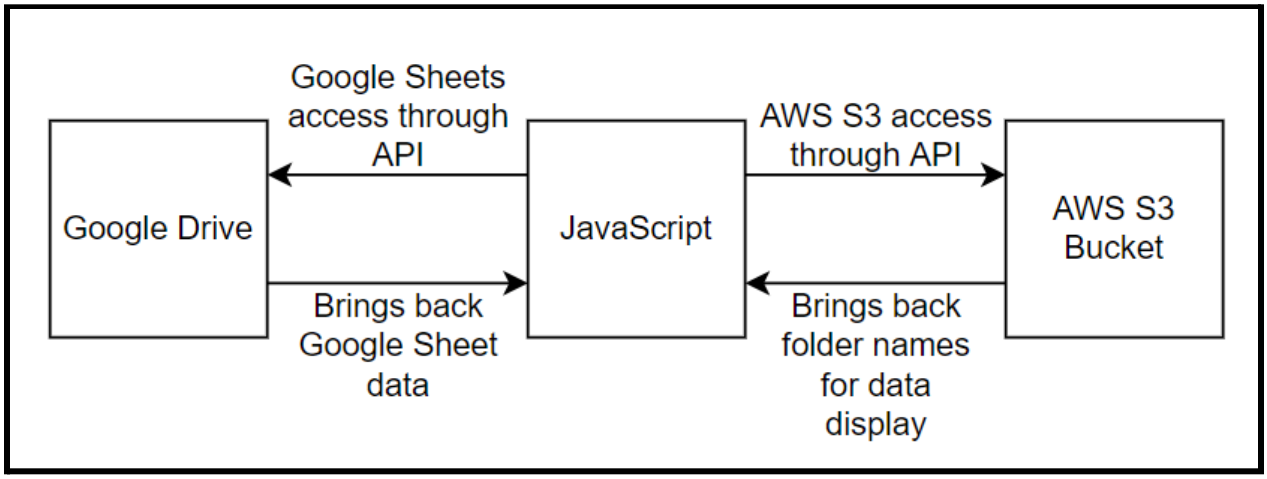
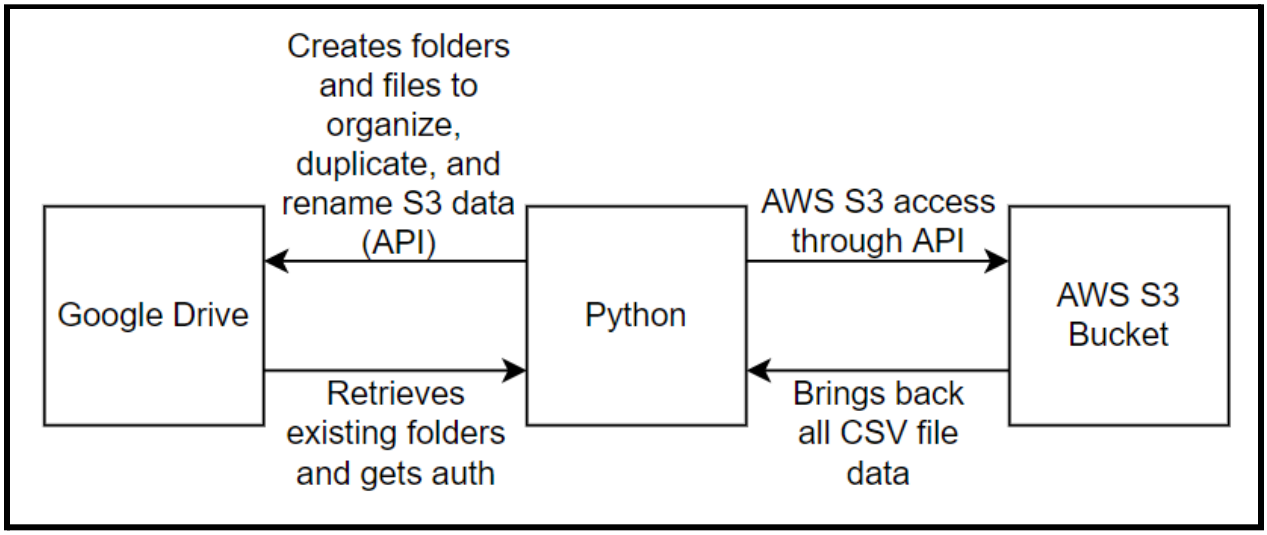


Figure 3: Python Architecture



Next is the frontend, which consists of Google Sheets and Wix. The reason that Google Sheets is considered the frontend is due to the data within the Google Sheets being necessary to populate graphs displayed on the site. The graphs populated are the main front-page graph on the site, as well as any graphs that show up in the hierarchy. Wix is the main system used as the front-end section of the site, as the website-builder contains all CSS and HTML inside of the site and out of the creator's view.

- **Python Scripts**
  'Main.py' collects CSV files from the necessary AWS S3 bucket, reads through each CSV file, recreates the CSV file to be pushed to Google Drive, creates folders for thorough organization on the Google Drive, and creates text logs.

‘Summaries.py’ also collects CSV files from S3 and reads through each of them, creating a summary for each exoskeleton, user, and CSV file that it comes across. ‘Functions.py’ only has the functions within the script for the other two scripts to access.

- **Google Drive**
  The Biomotum Google Drive is where all of the easily readable data from AWS S3 is stored. There are two main folders, ‘Documentation’ and ‘All Device Data’. The documentation folder contains important information about the site and logs that each Python script creates. This is to keep a user up to date on how to use everything and make sure the scripts are always running. The device data folder stores all of the CSV and summary data. Once in ‘All Device Data’, there are another two folders, ‘Users’ and ‘Devices’. The users folder contains folders of every user and all CSV data on that user. The devices folder contains every device, and then the users for each device, and the CSV data on the user and device.

- **JavaScript**
  The JavaScript is found within the Wix website’s frontend and backend. All JavaScript can be found there, simply being used to access AWS S3, Google Drive, and Google Sheets to find and display information. The rest of the JavaScript is used to make the site work and do things like find the list of exoskeleton a user needs, update databases, provide hierarchy names, and more.

Testing

Unit Testing

Software unit testing is the first stage of the software design process life cycle. Unit testing involves taking apart each of the major components of the project as a whole and breaking them down into their individual components to be tested with certain conditions. For example we can compare our web portal to a car. When testing our "car" we will split our car up into parts such as, wheels, engine, suspension, chassis, and so on. We then take each of those parts and ensure they meet the conditions they were designed to handle such as the doors properly closing, the engine delivering power and more. The general intent is to ensure that all parts of our codebase gets tested individually to ensure that they are all functioning properly. The goal is to ensure that no major outstanding defects are delivered when presenting our project to our client. We have split our unit tests into two groups including front end testing and back end testing.

Front End

The front end of our project is the face of our work, this is what the client and users will see and interact with at all stages of the interaction. It is vital that we ensure the front end is thoroughly tested to ensure that the user does not encounter any major errors with the interface. Our front end is Primarily composed of Wix tools and Javascript that is in charge of interacting with the buttons to achieve certain functionality. Our testing consists of testing the functionality of the Wix widgets, the buttons, the responsiveness, and the functionality of our javascript code, that is in charge of receiving the data from the buttons. We started with the beginning of our front end development functionality by initially testing our homepage. In order to test the functionality of the front end portion of our site we need to ensure that:

1. The front end correctly references and collects data from our database
2. That when the submit buttons is pressed, the proper database requests are sent
3. The front end button functionality remains reactive and consistent
4. The graphical displays stay consistent error free
5. How fast can we load the information we are looking for?

Back End

The back end of this project is where all of the major connections exist so we need to ensure that the back end functions are working properly so that correct data is displayed to the user. This is where all important calculations and functions are stored that allow for the front end to display accurate statistics and graphs. The back end has been tested by running and testing

each individual function from the four back end scripts we have in our project as well as testing the databases to ensure they are updating correctly. More specifically, in order to test the our back end portion of our site we need to ensure that:

1. All individual functions within the “google_sheets.jsw” script are working properly
2. All individual functions within the “googledrive.js” script are working properly
3. All individual functions within the “other.jsw” script are working properly
4. All individual functions within the “s3.jsw” script are working properly
5. The “Members” Content Manager database is updating and storing user data correctly

Integration Testing

Integration testing requires testing modules that work together at any point of application use. It is necessary to make sure that the application modules do not have any errors while working together, and that each module interacts/reacts in the appropriate way when affected by another module. While the team determined which modules should be tested and how, we based it on the characteristics above. Modules in the project that work with other modules were chosen and have been tested by triggering every interaction possible between the modules and watching for errors and/or the correct outcomes.

Some modules created during the time given for this project are not testable for the application, such as the Python script that organizes and transfers the Biomotum data files to the Biomotum Google Drive for employees to access and better understand. Instead, the script contains code that creates change and error logs to make sure all files are placed in the correct area and that all errors are tracked so they can be fixed. Most of the functions used within the script work together and can be tested using Python’s library unittest. While unittest has a name that seems to be unit test specific, the library can be used for integration testing practices as well. For every function that uses a different function within the code, a specific input to the first function used is set as well as the expected outcome. Then, the set input is used as the first function’s parameter, and the returned data is compared to the expected outcome. If they are the same, the integration test is successful, if not, it fails. This testing has been done for the Python script that connects to Google Drive, as well as the Python script that creates Biomotum exoskeleton and patient data summaries stored in the Biomotum Google Drive.

As for functions and modules implemented within the Wix website itself through the JavaScript language, a similar process as explained above has been used to go through integration testing. Although, instead of testing through code (like the Python unittest library), the tests have been through human interaction of the website.

To start off the testing, the Google Sheet involving the global summary of all exoskeletons and users have been manually changed to insure that changes are made on the

website, since the Google Sheet is connected to the site and should automatically be updated when there are any changes made. Then, the sign up function is tested to make sure that the new user's company name shows up in the admin's dashboard, where the admin can assign and unassign exoskeletons per user. Once the user is included in the dashboard, the user's patient page is tested by first making sure that the user has no access to any exoskeletons. Then, exoskeletons have been added to the user's patient page and checked by going to the page and going through each exo, user, and data sheet provided. The removal of access to exoskeletons has been tested by removing a few, and then all exoskeletons previously granted access to the user, and the patient profile of the user has been examined to make sure that there are no exoskeletons or users to access once again.

Lastly, the connection to each exoskeleton, user, and csv summary commences. This is in the same realm as the testing mentioned above, where access to certain exoskeletons exist. Although, this testing is to make sure that the information within each summary Google Sheet for the exoskeletons and users are correct. To do so, about 10 exoskeletons and users have been randomly chosen to compare and contrast the summary graph created within the Wix site to the summary sheets exported to the Biomotum Google Drive. The downloadable CSV (Comma Separated Values) file has also been compared to the file sitting within the Biomotum AWS account.

Usability

Usability testing is very important when it comes to making the client happy with the software being delivered. It focuses on the interactions that end users will have with the software system and the goal is to ensure that users are able to effectively use all of the functionality provided by the software. This is where we get feedback from the users to make any necessary user interface changes that will contribute to the ease of use of the software.

The web portal that we have developed tracks data from exoskeleton prototypes that are still being tested by Biomotum employees. Therefore, the usability testing has been conducted among multiple Biomotum employees to get their opinions on the design and usability of the web portal. This is due to the exoskeleton being in the developmental phase and has not gone out to clinics. The primary use case that has been examined was the end user or clinics. The end users were Biomotum employees who were given the role of a clinic working with the exoskeleton. In addition to the mentioned use case, we have developed a survey to help us understand exactly what the experience was.

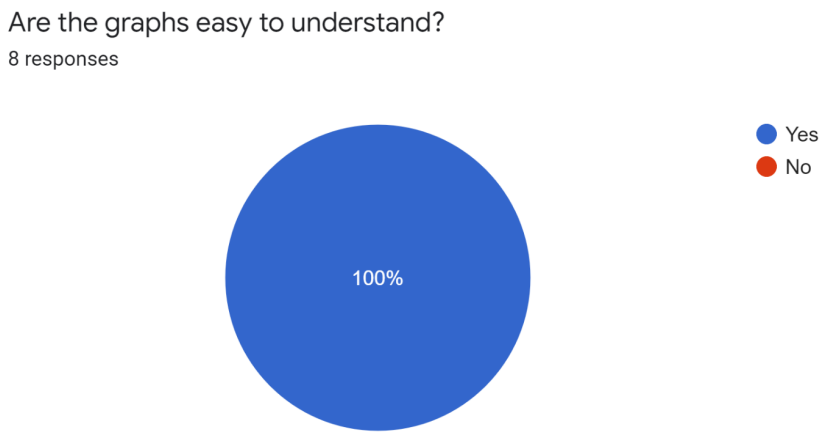
The first is the clinic survey which focuses on the account creation, viewing of data, and the downloading of the CSV files from the portal. This survey has covered design choices for the

website, aesthetic and overall look of the portal, the ease of finding said features, the understanding of the features available for both parties, and overall functionality of the features.

We chose surveys as our medium of testing because it allows for mass feedback on what was bad for both use cases. The regime had two groups of students for both use cases with both unable to interact with each other. The testers were given minimal instructions on how to create a new account under the company name of "Company Trial_Clinic_*clinic number*" and navigate to download a particular CSV file with a naming scheme of "Trial_Clinic_*clinic number*"- the clinic number has been provided upon the start of the trial. After both tasks are completed, the users were given the survey to fill out.

The survey results have provided us with lots of detailed feedback regarding each section of the use cases. This usability testing has helped to ensure maximum client satisfaction with the entire web portal. All of the results we received were very positive and the testers were able to successfully complete the test and survey. As you can see from figure 4 below, the results show that the graphs created were very easy to understand and there were no complaints about that. Although we did receive some complaints about minor bugs that resulted in the web portal loading inconsistently, all of those issues have since been resolved on both Wix's end and our end.

Figure 4: Sample Survey Results



Project Timeline

We have implemented many modules by having each team member focus on a specific module. This has greatly supported parallelism and allowed us to get the individual main modules of the overall project done quickly. This has also given us more time to integrate all of the modules, test the system and verify it with our client.

Diagram 5: Milestone Gantt Chart

Week	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Admin Data Visualization																
User Account Authentication																
Hierarchy Data Filter																
AWS pinging																
Downloadable Raw Data																
Admin Exoskeleton Selection																
Module Integration																
Bug Fixes																
Testing																

Diagram 5 shows the most up to date schedule for implementing the main modules of this project in a gantt chart. Many of the modules have been worked on at the same time since they have been split up for each team member to focus on one module. Everyone has had some input and contributed to each module but having each team member assigned to a module allowed us to get the project done quicker and more efficiently. This allows for everybody to have something to work on all the time and if somebody gets done early, they can always help with another module. We have fully integrated the modules as well as fixed multiple bugs throughout. We are currently in week 16 where we are finishing up documentation and writing the user manual for the client before we deliver the product.

Future Work

Even though we have delivered on all agreed upon requirements of our project, there are two main areas that can be used for future improvements. First, letting users assign exoskeletons would save a large amount of time for the administrator, second being preloading all primary data in the background in order to minimize load time from page to page without affecting the speed of the site for the user.

To focus more on the aspect of assigning exoskeletons. If the user was given the ability to accurately assign their own exoskeleton, the step that the admin would normally have to do (assign the exoskeleton manually to the user) would be eliminated saving tons of time. Also keeping in mind that this user base could grow exponentially, more and more time would be saved in the overall long run.

Keeping in mind the future stretch goal of making this website run as fast as possible for the user. One way to help improve the overall load time and the fast interactive feel would be to load data ahead of the user request. This is a common optimization technique that many developers use in order to make their tool of software seem extremely fast. You could implement a system that attempts to pre-render graphs before the user requests to see them, making it seem like the graph loads instantly the second the user requests the visual. You would also have to ensure that the “preloading” was something that is done when the site is static. It is vital that when the user is making a request (to see a page, load data, download csv) that all resources are dedicated to whatever their request is. When the site is static, meaning there are no requests currently, that time is essentially being wasted if you do not use that time to do something else. The static portion of the time is the time that you want to dedicate to loading extra information to make the site transitions more seamless.

Coming to our two future work ideas would improve the site from the perspective of both the user and the administrator. Future work and what would improve the website the most is highly subjective but we at rehab remote believe that focusing on saving the administrators time and improving the website feel for the user would be the most beneficial decisions for the site.

Conclusion

Biomotum has developed an ankle-exoskeleton that tracks the progress of patients with cerebral palsy and eventually assists others in need of rehabilitation. Our project will assist doctors and researchers involved with Biomotum by helping to understand the data that the ankle-exoskeleton collects. We have accomplished this by creating a web portal that continuously pings an AWS server which contains all of the ankle-exoskeleton data. The data will be sent and updated in the web portal where it is organized into various hierarchies including global data, clinic data, doctors' patient data, and trials. From there the web portal displays various charts that are easy to understand pertaining to the exoskeleton data as well as a button to download and save the raw data or charts.

There are many requirements in this project that have been met to ensure success of the project as well as client needs. Some functional requirements include a user authentication system that's secure, data visualization in the web portal, a data hierarchy that can be filtered to find specific sets of data, recurring AWS pinging, and the ability to download the raw data. All of these functional requirements have their own unique performance requirements that allow for the web portal to run smoothly and efficiently. There are also environmental requirements in the project that are outside of the development teams control like the dataset itself, and the web hosting platform. Furthermore, this project contains multiple potential risks that have varying severity levels including a patient data breach, unauthorized data access, and inaccurately displaying the data. The current team is proud of the result and delivery of the requirements and technologies that have been used to satisfy the requirements. The team is proud of the deliverable and wishes the best to Biomotum in the future.

Appendix A: Development Environment and Toolchain

Appendix A consists of information regarding the development environment of the system as well as the tools involved to create and maintain the system. The next sections will outline the hardware, tools, setup, and production cycle of the system.

Hardware

This system was developed on a variety of different hardware platforms, though most of the system is located within a remote AWS EC2 instance. The code was developed mainly on a system with a 10th generation Intel i7 processor and 16 GB of RAM. The instance that is constantly running all Python scripts has a Linux/UNIX platform with two virtual CPUs. We feel as though a computer with minimum specs of an i5 processor and 8 GB of RAM could be used for upkeep of the system, no matter the OS.

Toolchain

The only editor used to develop the Python scripts that belong within the AWS EC2 instance was Visual Studio Code, a Microsoft developed editing environment and debugger. This editor was used due to preference of the coder, not any other reason. The AWS EC2 instance holds the Python scripts that run constantly. AWS EC2 is considered cloud computing power by Amazon, and was necessary for the ability to run the Python scripts non-stop without having to use the power of a local computer. The Python language was used for certain scripts, as it is a preferred language to the coders and includes the necessary tools for the system such as pandas (an open source data analysis and manipulation tool that was used to access CSV files and create dataframes), pydrive (a wrapper library for the Google Drive API used to access and manipulate the Biomotum Google Drive), and boto3 (a Python API for accessing AWS infrastructures, necessary to access the AWS S3 bucket files).Wix (a website creator) was used for the CSS, HTML, and serving portions of the project. This was used since Biomotum already uses Wix to maintain their other website and they are familiar with the UI. FileZilla was also used to access the files within the EC2 instance.

Setup

There is no specific way to set up the environment to access the site. The project will be available on the domain that Biomotum chooses once the site is transferred to them. Though, some important files are included in the Google Drive documentation, as well as the EC2 instance. To access the EC2 instance, all that needs to be done is to ssh into the instance with the .pem file that is available on Google Drive. Make sure to ssh in the directory that the .pem file is stored in on the local computer, and use the main ssh tool on your OS and type ‘ssh -i "biomotum.pem" ubuntu@ec2-34-222-9-240.us-west-2.compute.amazonaws.com’. The

‘ubuntu@...’ could end up changing based on any changes made to the instance from here on out. The user attempting to connect to the instance only has to click the instance and the button ‘connect’ to see the new ssh line of code.

Otherwise, the administrators can access the website through Wix. This is necessary so that the admin can see the exoskeleton assignment area, which assists in assigning different exoskeletons to every user.

Production Cycle

To maintain the backend of the web portal, the user will have to access the EC2 instance using the ssh instructions mentioned above. They can use FileZilla to gain access to the files within the instance and easily copy over files to their local computer. This is done by clicking ‘Open the site manager’, adding a new site, changing the Protocol to SFTP, making the Host ‘ec2-34-222-9-240.us-west-2.compute.amazonaws.com’ and Port 22. Then change the Logon Type to Key file, make the User ‘ubuntu’, and Browse for the key file in the local computer to find the Biomotum .pem file. Once that is done, click connect, and all access will be completed. It will then be seen that the scripts sit inside the root folder, so there is no need to look through folders to find them. Some other files such as .json files and .yaml files also sit within this directory. Do not touch those. They are used to connect to the Google Drive API within the scripts. If scripts are changed, drag the changed scripts to the FileZilla window to update the files in the instance. Then, ssh into the instance. If you want to re-run the summary script, type ‘tmux attach-session -t summaries’, type ctrl+c, then type ‘python summaries.py’. If you want to re-run the main script, type ‘tmux attach-session -t main’, type ctrl+c, then type ‘python main.py’. The tmux is so each script can run at the same time. To leave these tmux windows, simply type ctrl+b, then d.

To maintain the frontend of the web portal, the user only has to login to Wix and edit the Biomotum portal site. Each page has the frontend JavaScript code displayed and shows the backend JavaScript being imported to the code. To test the site, make some changes, then click ‘Run’. This will not publish the site, only test it with the changes made. To publish, simply click ‘Publish’. The backend code can only be accessed using Dev Mode. Once in Dev Mode, go to ‘Public & Backend’ to see all code used in the backend portion of the site. In addition, there is the ‘Databases’ section. If clicked, this will show the databases available for the site. ‘Members’ is only the database that matches members with exoskeletons. Lastly, in ‘Developer Tools’ there is ‘Secrets Manager’ which holds the velo Google Drive, Google Sheet, and main Google Drive folder ID credentials.

References

[1] Centers for Disease Control and Prevention. (2020, December 31). Data and statistics for Cerebral Palsy. Centers for Disease Control and Prevention. Retrieved November 5, 2021, from https://www.cdc.gov/ncbddd/cp/data.html.

[2] Science & Outcomes. Biomotum. (n.d.). Retrieved November 5, 2021, from https://www.biomotum.com/science.