Milestone 3

Title:

SmartStride: Toe-Walking Rehab

Names & Emails:

Cianna Grummer cgrummer2019@fit.edu

Alec Anzalone aanzalone2021@fit.edu

Kiera Ceely kceely2021@fit.edu

Bela Perdomo iperdomo2021@fit.edu

Caleb Phillips cphillips2021@fit.edu

Faculty Advisor:

Dr. Gu gul@fit.edu

Progress for Milestone 3:

Task	Progress	To Do
Connect the logins to individual patient and practitioner pages	100%	N/A
Add patient button to practitioner page	100%	N/A
Add patient page	100%	N/A
Add patient functions	100%	N/A
Use “Dummy” data to be displayed to patient and practitioner	100%	N/A

Discussion:

- Connect the logins to individual patient and practitioner pages:
    - The Clinician and Patient pages now when logged in pull the user data from the database using the created API resource and lambda function to display the user data to the user personalizing the pages. The two clinician and patient pages are two separate html files that act as a blueprint for the user data to be inserted into. Upon logging into the patient portal the patient dashboard will send two different API calls off. One is to retrieve the “Dummy” data witch will be explained in a later section and the other is to retrieve all the user profile data. When sending for

this information the username of the user that had just logged in is sent as a POST method body that the API then sends to the GetPatientData lambda function. Upon receiving the body from the API, the lambda will connect to the database and search the patient table for a patient with a username that matches the one sent in the body. If no user is found the error message will be sent back to the API but if a user is found all the data stored in the table except for the password will be extracted and sent back to the API as a JSON body and the API will sent the information back to the patient dashboard html where it will parse the JSON body and use the information received to assign the variables to a value that is used to display the patient's information. The process for a clinician works the same way except the patients list is extracted as a list within the list of user data. The same lambda is called upon when a clinician clicks on a patient to view their page except when a clinician accesses the patient's page many sections of code are activated allowing the hidden graphs, buttons and editabilities to be viewed only by the clinician.

- Add patient button to practitioner page:
    - Within the practitioner's page they can add or remove patients from their patient list. When the doctor clicks on the three dots to the right of every patient they can click remove the patient. If clicked this will give the doctor a prompt asking if they are sure they want to remove the patient. If yes, the API will be invoked sending the username of the doctor and the patient to the lambda RemovePatient. When the lambda receives the information, the lambda will connect with the database and search in both the patient and doctors' tables to find the users. When found the doctors list of patients will be edited to remove the patient and the patient in the patients table under their assigned doctor will be deleted. The patient no longer has a doctor so they will be a floating patient. This brings me to the Add new patient in the doctor's page. The doctor will select the button to add a new patient and be taken to the add patient page.
- Add patient page:
    - A new page is needed to hold the form to enter the patient that the practitioner wants to add. The form will take in the username of the patient they want to add and when the doctor gives the patient's username to be added to the list of their patients it will invoke an API call that will give the username of the doctor and patient to the lambda. The lambda will then connect to the database, search for both the doctor and patient within the respective table and if the patient does not have a doctor already, they will be added to the patient list for that doctor and the doctor's username will be added to the patient's doctor.
- Add patient functions:
    - Five API resources were needed to be created all with POST methods. The APIs that were created are called: GetPatientData, GoalHandler, ITW_GetCSV, ITW_

RemovePatient, ITW_AddPatient. For each one of these APIs in the POST method I created a Method Request that handles the JSON body being sent and received from the html and lambda, the integration request was set up with the corresponding lambda function by using the endpoint of the POST method in each resource and adding that to the lambda as well as the html acting like an 'address' of where the API resource is, set up the integration response so the website has permissions to access that API resource method by editing the access-control-allow-origin, then lastly enabling cross-origin resource sharing or CORS. CORS allows the virtual private cloud or VPC to access and use the API recourse method then finally deploy the API.

    - Five lambdas were created, and I will list them in order of association to the API resources that were created: ITW_GetPatientData, ITW_GoalHandler, ITW_GetCSV, ITW_RemovePatient, ITW_AddPatient. For each of these lambda functions A VPC needs to be assigned as well as the proper security groups and subnets establishing the outbound and inbound rules. The IAM roles need to be edited after adding the VPC to allow the lambda to use the VPC assigned by giving the IAM role of the lambda permissions to use the VPC. Then a CloudWatch log needs to be created to allow the output of the lambda to be received. Then appropriate layers need to be imported and assigned to the correct lambda. Then I code the appropriate JavaScript code for the actions of the lambda to execute. Then lastly, I assign the endpoint of the corresponding API to the lambda's API Gateway trigger and deploy the API.

- Use "Dummy" data to be displayed to patient and practitioner:
    - My group provided me with some raw test data that I was able to use to create a new table in the database called patients_session_data that holds EMG, accelerometer 1 xyz, accelerometer 2 xyz and Gyroscope 1 xyz, Gyroscope 2 xyz data. The data has a session ID as well as a patient ID and when a patient or a doctor logs into a patient dashboard the patient's username is sent to the API which send the json body to the lambda function that will connect to the database then searches for the patients in the patient_session_data and pulls every 10^{th} line of data from the same session, reduces the noise of the data then sends in back to the patient dashboard html. Once received the data is filtered and turned into a graph and displayed on the profile page but this process can take a moment to go through all the data because the data that is being processed is over 11,000 rows of data so a loading wheel is displayed while the data is processed into a graph.

Contribution:

I have created all the functions, API calls, API resources and methods, Lambda functions and database connections, as well as creating all the VPC security groups that are needed to

complete the functions myself. Bela from SmartStride is providing sensor data that I can use to build the framework of how the data is handled and displayed to the clinician and patient.

Plans for Milestone 4:

Task	Progress	To Do
Develop a step identification method	10%	Identify how many steps were taken for the duration the device was worn. Signal Processing
Create A Percentage of ITW step function	20%	Create the framework of how the percentage will be calculated and displayed
Create respective API and Lambda functions	0%	For all the mentioned functionalities create the Lambdas and API resources
Create severity bar functions	0%	After developing step identification method and Percentage of ITW steps use results to update the severity bar

Discussion (Future Milestone):

- Develop a step identification method:
    - I will need to research and learn some signal processing in order to process and cut the data into individual steps. To do this I will need to be able to detect footfalls by peaks in vertical acceleration and identify periodic oscillations that will correspond with the gait cycles. Then I will need to detect when a specific muscle is activated using the EMG data. Different muscles are active during different stages of the gate cycles. Knowing this information, I can take the data that Bela has provided me with and do low-pass or band-pass filtering. Since I am still learning about this subject, I am not sure which filtering method will yield the best results but that is one of the topics I will be researching for this task. Among many more.
- Create a percentage of ITW steps function:
    - After the data is segmented by gait cycle, I will try to classify the steps as ITW steps and Normal steps using these criteria:
        - No/ weak heel strike detected = Toe walking
        - Prolonged front foot contact = Toe walking
        - Overactive muscle use = Toe walking

    - Using these criteria, I will try to separate the number of steps that are ITW from normal steps and count how many ITW steps the patient made and compare it to the total amount steps taken in total to give a percentage of ITW steps. From this ITW classification I will be able to build upon this in future milestones to enhance the graphs to better show doctors and patients the patient's information in a more digestible way and be able to show the information in a format other than a graph.
- Create respective API and lambda functions:
    - For both of the above tasks I will have to create new lambda and API calls in order to process the data server side then return the results to the API that will send it as a POST method to the html. I have already described how I set up lambda and APIs in a previous discussion and encourage you to read it to understand the steps for setting them up.
- Create severity bar functions:
    - After I am able to process the data and break it up into the gate cycles and then identify if the step was ITW or normal step I should be able to classify the severity to a certain degree. I will be working heavily with my group on how they would like to classify the four severity levels and what patient states indicate what level of severity that patient is. Once this is determined I will be able to categorize the patient and update the severity bar on their profile.

Meeting Dates:

Wednesdays 1pm-2pm

Client Feedback:

Group Feedback:

- Display the Names instead of Usernames
- The format of the patient data collected by the device may change from CSV files to graphs or something else to be determined later
    - Keep the data input versatile/ flexible
- Alec is working on creating the graphs
    - Help Alec by developing a step identification method
- Add Gate analysis, foot angle analysis, and EMG analysis to the website sections of the patient dashboard

Dr. Chan's Feedback:

- Discuss how to present the data to the doctors as they might not know how to read the data graphs

- Determine what a doctor would find helpful from our collected data when diagnosing a patient with ITW
    - Talk to the BME group about this
- Possibly include a timeline of when frequent ITW steps are taken
    - Knowing when the ITW steps are most frequent might help the patients in their rehabilitation process

Advisor Meetings:

Discussions happened over email.

Advisor's Feedback:

•

Faculty Advisor Signature: _____ Date: _____

Evaluation by Faculty Advisor:

Task for Faculty Advisor: detach and return this page to Dr. Chan (HC 209) or email the scores to pkc@cs.fit.edu

Score (0-10) for each member: circle a score (or circle two adjacent scores for .25 or write down a real number between 0 and 10)

Cianna Grummer	0	1	2	3	4	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5	10

Faculty Advisor Signature: _____ Date: _____