SW Engineering CSC 648/848

Team 2

|  |  |
| --- | --- |
| Version History | Date |
| M2V1 | 20OCT2020 |



SFSU | CSC648/848

Milestone 2 Documentation

**Team**

Team lead/Scrum Master:

Jarett Koelmel; [jkoelmel1@mail.sfsu.edu](mailto:jkoelmel1@mail.sfsu.edu)

Front-end Lead: Chiu Wong

Front-end Dev: Michael Canson

Front-end Dev: Paul Borst

Back-end Lead: Peter Hu

Back-end Engineer: Brooke Porter

Git Master: Eric Chen

Fall 2020

Table of Contents

[1. Data Definitions V2 2](#_Toc53332001)

[2. Functional Requirements V2 8](#_Toc53332002)

[3. UI Mockup and Storyboards 9](#_Toc53332003)

[4. High-Level Architecture & Database Organization 10](#_Toc53332004)

[5. High-Level UML Diagrams 11](#_Toc53332005)

[6. Key Risks 12](#_Toc53332006)

[7. Project Management 13](#_Toc53332007)

# Data Definitions V2

**Unregistered Users**

An Unregistered User is a user that does not currently have an account or is not logged into the system. Their usage of the website is limited to the landing page which displays information about the application and company and the registration page so they can create an account if needed.

**Functionality**

* Can see general info on the landing page and access registration
* Cannot access therapist dashboard, listings of other therapists, exercise library, or any information about patients that is shared between other registered users

**Registered Users (Therapists & Administrators)**

A registered user is a user that has an active account and has had their credentials authenticated in order to be considered logged in. Users are largely physical therapists, with some administrative personnel mixed in. The capabilities of these two use cases overlap significantly. This table will provide all necessary

**Functionality**

* Can access therapist dashboard which provides instant insight to messages, current and potential client lists, and custom workout playlists for assignment
* Can access directory of other therapists for purposes of sharing information and patient-submitted videos, when permitted by patient
* Can access patient progress logs of patients directly associated to the user
* Can view messages generated by patients associated with their user account
* Cannot see metrics and statistics of other users, unless user has administrative permissions

**Attributes**

* User\_ID(Primary Key): int
* Email: string
* Password: hashed-string
* Name: string
* Address (Composite): string
  + Handled via a separate table so references of many therapists in one office can be handled by one data entry, attributes broken into elements (house number, street, city, state, zip code, etc)
* Company: String
* Specialization: [Injury/Area]
  + This can be either specific injured areas (back, knee, shoulder) or specific area of expertise (sports injury, elderly rehabilitation, pediatrics, etc)
* Patients: [patient\_id]
  + Used by the foreign key in the patient table to associate therapists to patients under their care. 1:M relationship
* Admin: boolean
  + Permits checking the user’s permissions so that additional functionality can be afforded to those that require it.

**Administrator**

An administrator is a child class of registered user in that all of the functionality of the registered user is inherited but the additional functionality is discrete from traditional users. All of the attributes will be the same for the administrators as it is for the therapist users, with the key difference being that

**Functionality**

* Can see the tracking statistics of all therapist accounts under their supervision
* Can run reports of therapist statistics over different time scales for performance assessments
* Can see patient progress reports of all patients associated to their therapists for purposes of quality control and client satisfaction
* Can remove therapist accounts

**Attributes**

* Admin\_ID(Primary Key): int
* Subordinates: [Therapist]
  + This will contain a list of all therapists under a specific administrator’s supervision

**Patient**

A patient is a user that has an account and is logged into the patient portal. While this is a discrete element from the application this team is building, their data is essential to the proper operation of our therapist portal.

**Functionality**

* Can directly message therapists currently assigned to patient
* Can view the directory of therapists in the system to assist in finding a suitable therapist that meets the patient’s needs for rehabilitation
* Can upload videos of home exercises performed for therapists to review and provide feedback on
* Can view feedback on videos uploaded to their therapists
* Can grant permission for their personal videos to be shared to other therapists/patients

**Attributes**

* Patient\_ID(Primary key): int
* Name: string
* Address (Composite): string
  + Handled via a separate table so references of many therapists in one office can be handled by one data entry, attributes broken into elements (house number, street, city, state, zip code, etc)
* Injury: [Injury]
  + This can be used to filter therapists that provide services for their injuries or specific needs
* Therapist: int
  + This will be a foreign key to relate this data entry to the associated therapist
* Assignment: [Exercises]
  + This will be used to point to the exercise regiment created by their therapist
* Progress: [Log entry] (string)
  + Each progress entry from the therapist will be appended to this log entry array.

**Exercise Library**

The exercise library will contain all videos that can be utilized by therapists on the system. The ability to augment this collection of videos is afforded to the therapists via their video upload tools within the exercise library page. This will allow for crowd-sourcing of more specific exercises that are required beyond the initial offering by the application.

**Functionality**

* Can be searched based on the tagged body parts so that therapists can find specific exercises and videos faster
* Can be added to a workout playlist which can also be stored in a separate table that allows for quicker referencing of already constructed workouts
* Can be appended by therapist-uploaded videos for non-extant exercise videos

**Attributes**

* Video\_ID(Primary Key): int
* Video\_altText: string
* Length: int
* Description: string
* Tags: [string]
  + This array of tags will be to filter search results and return those applicable to the therapist

**Patient Video Library**

The patient video library will contain videos from all patients that have been uploaded to the system. The videos will be connected to their uploader and this will prevent other therapists from viewing them unless they are the associated therapist for that patient. The option to share videos does not immediately release them for viewing by anyone, but relies on therapists to share the videos at their discretion.

**Functionality**

* Can be shared at the individual level, as dictated by the uploader.
* Can only be accessed by proper users
* Cannot be searched for obvious privacy reasons

**Attributes**

* Upload\_ID(Primary Key): int
* Upload\_altText: string
* Length: int
* Description: string
* Shared: boolean
  + This will be decided at time of upload in the patient upload by a checkbox that will allow for their therapist to utilize their video beyond assessment purposes

**Custom Workout Library**

The custom workout library is primarily tied to each therapist as they create workouts for their patients to perform and assign them accordingly. The ability to share workouts previously made is key to simplifying therapist productivity system-wide.

**Functionality**

* Can contain references to numerous exercises and associated descriptions that detail to the patient what needs to be done, how often, and how many times (repetitions)
* Can be shared between therapists at their discretion
* Can be renamed as needed

**Attributes**

* Workout\_ID(Primary Key): int
* Title: string
* Exercises: [Video\_ID]
  + This array of Video\_IDs will point the Workout to the correct exercise videos
* Descriptions: [String]
  + Array pairs with video IDs in order and lets the patient know what is to be done for each assigned exercise

**Messages**

The messages are a collection of encrypted messages that can only be seen and accessed by the patient/therapist pair involved in creating the communication. This direct message system will allow for asynchronous, around-the-clock, connections to be made between patients and therapists to suit their needs flexibly.

**Functionality**

* Can only be sent between patients and therapists associated with one another
* Exists on the system for the duration of the patient’s active status on the system, after which, can be deleted at the discretion of the therapist

**Attributes**

* Message\_ID(Primary Key): int
* Message: hashed string
* Patient\_ID: int
* Therapist\_ID: int
  + Both therapist and Patient IDs are used for ensuring recipient and can be built into hashing algorithm so that the unique IDs accessing them affect encryption and ensure privacy

**Progress Log**

The progress log is a collection of inputs from the therapist that details a particular patient’s progress throughout their time in therapy. This provides a sortable history of rehabilitation process and is viewable by the therapist and any administrator assigned to that therapist.

**Functionality**

* Can be sorted by date and time of creation
* Can be appended to by therapist in charge of patient’s therapy
* Can be viewed by administrators in charge of therapist that created the entries

**Attributes**

* Log\_ID(Primary Key): int
* Log: [Entry]
  + This is an array of entries that will be generated as the therapist creates them
* Patient\_ID: int
* Therapist\_ID: int

**Progress Entry**

A progress entry is the unit contained within the larger data structure, progress log, it will have pertinent information about the patient’s status and progress as detailed by the therapist at the time of creation.

**Functionality**

* Can be created and edited by therapist
* Can be viewed by administrator
* Contains patient information related to rehabilitation progress

**Attributes**

* Entry\_ID(Primary Key): int
* Entry: string
* Created\_on: date-time
* Log\_ID: int

# Functional Requirements V2

# UI Mockup and Storyboards

# High-Level Architecture & Database Organization

# High-Level UML Diagrams

# Key Risks

The following is a risk assessment of the development team in its current status:

1. **Skills**

Our team has a diverse set of skills that cover most of the requirements to fully develop this project. However, it seems that the skills are largely segmented between individuals and the minimal overlap may result in issues with workload balancing in the future as development progresses and becomes more intensive. In order to rectify this, it would benefit the team to cross-train or study with individuals that have skills they do not possess, if time and schedules permit.

1. **Schedule**

As a fully remote team, our scheduling cannot revolve around on-campus time for large development meetings or sessions. Coupling this with the extant work schedules of our teammates, the limited times available to have full-team meetings forces the team to rely more heavily on asynchronous communications, which is not preferable as we enter later stages of development and time-sensitive milestones. This presents a significant risk currently, but one our team is seeking to remedy before it becomes more disruptive. The best approach our team is currently undertaking is to break out into smaller teams to work on specific areas when people are available to meet, this has helped facilitate development outside of regularly scheduled meetings.

1. **Technical**

The current technical challenges revolve around adapting open-source AI tools to operate efficiently without our current frameworks and software stack. The most challenging aspects for development of the final product involve the optional but highly-desired functional requirements that facilitate faster assessment of patient-uploaded videos. The AI component presents a large computational requirement that may directly affect the end-user experience if not implemented correctly. This is certainly the largest technical risk currently facing the team.

1. **Teamwork**

The current environment which does not allow for direct interpersonal collaboration presents a significant risk to the team. This is exacerbated by the fact that the entire team is made up of people that have never worked together before, nor met in person. Bridging this gap, our team attempts to utilize Zoom, Slack, and GitHub to its full potential to maintain open lines of communication and meet all objectives in a timely and organized fashion. However, this is not to say that risks are not present when looking forward into further development.

# Project Management

Current development surrounding milestone 2 has relied on initiative undertaken by frontend and backend team leads to delegate tasks that fall within their scope to team members and flesh out granular details. As smaller deadlines and modules are completed, requirements and technical details are compared between the two teams for incongruency and issues are rectified during team meetings to reassess the team’s direction and path moving forward so that the team is moving towards the milestone goals as a cohesive unit.

Looking forward to future tasks and development, adhering to a flexible and adaptive strategy that is inherent with agile software engineering will allow for our team to pivot as necessary to meet changing demands from our client and address technical hurdles as they present themselves. Addressing the issues and errors that are bound to happen will be handled via blameless post-mortems so that the team can use these moments to learn and better themselves as engineers and prevent the weaponization of errors to embarrass or deride other team members which is notoriously bad for morale and teamwork.