mHealth Exercise Intervention in Pulmonary Arterial Hypertension

Pulmonary arterial hypertension (PAH) is a disease in which the small blood vessels in the lungs become obstructed. As a result, the right side of the heart, which pumps blood into the lungs, begins to fail. The disease worsens over time and about 50% of patients with PAH die within 5 years. Current medications have little effect on survival and cost from $20,000 to $175,000 per year. More effective and affordable therapies are therefore needed.

One potential treatment that has little or no cost is exercise. Previous studies have shown exercise to improve the condition of patients with PAH, but they involved rigorous exercise routines that are hard for patients to follow outside of a study. [App name here] is an app designed to help medical providers (such as doctors, clinicians, and/or researchers) develop a practical exercise intervention using information gathered from wearable activity monitoring devices. Patients participating in the study will wear an activity monitor on the wrist for twelve weeks. The monitor will have a display that can provide real-time feedback allowing, patients to monitor their progress toward daily goals. Patients will regularly transmit activity data from the device to smartphone or computer via a suitable communication protocol. Data will be available to doctors in real time.

Basic Project Requirements

Any Capstone project must support multiple users and should leverage services running remotely in the cloud. Each project's specification clearly outlines the app's intended high-level behavior, yet leaves substantial room for individual creativity. Students will therefore need to flesh out many important design and implementation details. Basic requirements for all Capstone MOOC project specifications include:

- Apps must support multiple users via individual user accounts. At least one user facing operation must be available only to authenticated users.
- App implementations must comprise at least one instance of at least two of the following four fundamental Android components: Activity, BroadcastReceiver, Service, and ContentProvider.
- Apps must interact with at least one remotely-hosted Java Spring-based service over the network via HTTP/HTTPS.
- At runtime apps must allow users to navigate between at least three different user interface screens.

- e.g., a hypothetical email reader app might have multiple screens, such as (1) a ListView showing all emails, (2) a detail View showing a single email, (3) a compose view for creating new emails, and (4) a Settings view for providing information about the user's email account.

- Apps must use at least one advanced capability or API from the following list covered in the MoCCA Specialization: multimedia capture, multimedia playback, touch gestures, sensors, or animation. In addition, experienced students are welcome to use other advanced capabilities not covered in the specialization, such as BlueTooth or Wifi-Direct networking, push notifications, or search, as long as they also use at least one advanced capability or API from the list above. Moreover, projects that are specified by commercial organizations may require the use of additional organization-specific APIs or features not covered in the MoCCA Specialization. In these cases, relevant instructional material will be provided by the specifying organization.
- Apps must support at least one operation that is performed off the UI Thread in one or more background Threads or a Thread pool. Communication between background Threads and the UI Thread should be handled by at least one of Android concurrency frameworks, such as the HaMeR or AsyncTask framework.

There may also be additional project-specific requirements (e.g., required use of a particular project-specific API or service).

Basic Functional Description and App Requirements

1. The *Patient* is the primary user of the mobile app. A *Patient* is represented by a unit of data which contains the core set of identifying information about a patient including (but not necessarily limited to) a first name, a last name, a date of birth, and a medical record number.
2. The *Patient* will be given a target activity level based on their baseline assessment (e.g., number of steps to walk, total activity time, etc.).
3. The *Patient* will receive a *Reminder* in the form of alarms or notifications during prolonged sedentary periods, up to 3 times per day during waking hours (9AM to 8PM).
4. The patient will complete a daily questionnaire. Questions will include:
    - What kind of activity did you participate in yesterday? (select all)
        - Examples include walking, jogging, mall/grocery shopping, light housework, yard work, weights, etc.

- What hours were you largely inactive yesterday, e.g., in bed, sitting for long periods of time, etc.?

5. A *Doctor* is defined as a different type of user with a unit of data including identifying information (at least first name, last name, and a unique doctor ID) and an associated list of *Patients* that the doctor can view a list of. A *Doctor* can log in.
6. A patient's *Doctor* is able to monitor a *Patient*'s daily activity level with data displayed graphically. The data is updated at some appropriate interval (perhaps when the daily questionnaire is completed).
7. A *Doctor* can see all his/her *Patients* and search for a given *Patient*'s data by their name.
8. A *Doctor* is alerted if a *Patient* does not meet the activity level target for 2 consecutive days.
9. A *Patient's* data should only be accessed by his/her *Doctor*(s) over HTTPS.

Implementation Considerations

- How will activity level data be transferred from a *Patient*'s device to his/her *Doctor*?
- How will data be stored on the server-side so that *Patients* are associated with *Doctors*? You may use a hard-coded set of patient and doctor user accounts provided by an in-memory UserDetailsService. You may also hard-code the relationship between *Patients* and *Doctors*.
- What will the user interface look like for a *Patient* so that it is quick and simple to use?
- How will *Reminders* be delivered to a *Patient* in a way that will help the *Patient* use the app more frequently and consistently?
- How will a *Patient*'s data be securely transferred to the server?
- How will a *Doctor* be able to update medication lists for a specific *Patient* and how will these updates be sent to that *Patient*'s device?
- What will the user interface look like for a *Doctor*?
- How, when, and how often will the user enter their user account information? For example, will the user enter this information each time they run the app? Will they specify the information as part of a preference screen?
- What user preferences can the user set? How will the app be informed of changes to these user preferences?
- How will the app handle concurrency issues, such as how will periodic updates occur - via server push or app pull? How will search queries and results be

efficiently processed? Will the data be pulled from the server in multiple requests or all at one time?

- How will the app use at least one advanced capability or API listed above? For example, will you create an animation to explain the app? Will you allow patients to take pictures of their weight gain/loss over time? Will you use push notifications to prompt users for information about their activity level? Will you allow users to employ simple gestures to snooze *Reminder* prompts for a set amount of time?
- Does your app really require two or more fundamental Android components? If so, which ones? For example, this app might benefit from using a ContentProvider or from using a background Service that synchronizes local and remote data, only when the device is connected to a WiFi network.