TASK ANALYSIS

As a member of the Deaf/Blind community, I have ways of finding a route to my destination, but lack a good way to know when my bus has arrived at the stop and would like a way to let the bus driver know which bus I am waiting for without carrying an extra notepad.

Task 1: Buzzing to a user at the stop when their bus has arrived

- The user defines which bus they wish to take and which stop they are starting from
    - The user defines (in a dropdown or similar input setup) the bus number they wish to take and the direction.
    - The user selects their initial stop from a drop-down (or select “Use My Location”)
        - Pre-Condition (for “use my location”): The user has (at some point) enabled Location permissions for the app.
    - The user selects their destination stop from a drop-down.
    - The user receives a buzz when the bus is some time/distance away.
- The technology notifies the user when the given bus is at the given stop.
    - The technology will use an API to get real time information about when the bus is expected to arrive.
    - The technology will buzz when the bus is some time/distance away.
- This task will be performed every time the user is expected to begin a new journey. This task would be performed in the environment of a bus stop or at home. The safety/noise levels of the environment depends on where the bus stop is located.
- Resource constraints:
    - Need data/wifi to get real-time data.
- What can go wrong:
    - Can lose data/wifi connection would lead to not knowing where the user is/not knowing where the bus is, but this can be handled by letting them know that connection has been lost through a buzz and some text readable by braille readers.
    - Location services are turned off would lead to us not being able to find and use their current location to detect which stop to use, so they would have to manually select the stop.
    - Our real-time data might not be fully accurate so we may be providing incorrect data. We cannot always know when this is happening, but can try to mitigate it by asking the user to confirm when their bus has arrived and they are on it.

Task 2: Displaying the bus number a user is waiting for

- The user defines which bus they would like to take to their destination
    - Same steps as 1a
- Technology shows the bus number the user entered.
    - The technology will display the number on the screen

- Bus driver looks out for this display screen and gets off the bus to help the individual get on.
    - Seattle Metro drivers notice the numbers displayed on the user's phone screen and offer assistance to the user
- This task occurs in the same context as task 1.
- What can go wrong:
    - The bus driver doesn't see the displayed bus number and thus will not get off the bus to help the user get on.
    - In the case that the user is in a dangerous neighborhood, they might take out their phone and hold it up while someone can just steal it away from them. In this case, it is the user's choice to display the phone or not.

As a member of the Deaf/Blind community, I have no good way of knowing when to get off of the bus that I am riding on, and would like to have a hands-free way to be aware of my relative position to my destination.

Task 3: Buzzing to a user on a bus when they are at/near their destination stop

- The user defines what bus they are on and which stop they are headed towards.
    - All of the steps for 1a
    - After the bus has arrived (and the app notified the user of this), the user steps on the bus
    - The user displays their phone screen to the bus driver, the screen displays a message like "Can you confirm that I am on bus XXX in direction YYY?"
    - The bus driver clicks a button on the phone screen to either confirm or deny the message
        - If the bus driver confirms, the screen displays "Thank you :)"
        - If the bus driver denies the message, the screen displays "Please help me off the bus"
        - The user receives a buzz when the bus is some time/distance away from the destination
- The technology notifies the user when the given bus is 2, 1, and 0 stops away from the destination.
    - The technology will use the bus number that the user entered along with their boarding time to figure out which bus the user is waiting for
    - The technology will use that information to track the bus using an API and check how many stops are left before arriving to destination
    - The technology will buzz when the user is 2, 1, 0 stops away from destination
- What can go wrong:
    - The bus driver doesn't see the buttons on the user's phone, this leads to the app not knowing whether the user is on the bus or not and can be resolved by the user reminding the driver, or pressing the buttons themselves.
    - The bus driver or the user accidentally press the wrong button (or the right button at the wrong time)

        - If the “confirm” button is mistakenly pressed (and in general when the confirm button is pressed), the next screen will offer a “Go Back” button to return to the confirmation screen
        - If the “deny” button is mistakenly pressed, the confirmation screen will pop back up within a few seconds and the mistake can be corrected
    - The app loses wifi/data while the user is on the bus, this leads to not knowing how close the bus is to the stop. We can alert the user that we have lost data with a buzz and some text, and also let them know how many stops were left last time the app had data (by text).

Task 4: Buzzing to a user when they are halfway to their stop

- Technology senses where the user is in their route and alerts the user that they are halfway through their route.
    - The technology will use the bus number that the user entered along with their boarding time to figure out which bus the user is on
    - The technology will use that information to track the bus using an API and check how many stops are left before arriving to destination
    - The technology will buzz when the user is (# of total stops in their route / 2) away from destination.
- This task is performed on the bus. The level of noise will depend on how crowded the bus is.
- What can go wrong is same as for task 3.

Task 5: Allowing user to define when to receive buzzes/which buzzes to correspond with what information

- The user defines which number of stops away to receive a buzz for
    - The user clicks a button to “add a new alert”
    - The user selects whether the alert is for a number of minutes (until the destination) or a number of stops (away from destination)
    - The user a number from a provided number drop down
    - The user selects which type of buzz from a “buzz-builder”
        - Buzz Builder has 2 options for a buzz, a long buzz and a short buzz
        - User clicks the corresponding button to add a buzz to the pattern, thus building their own buzz pattern for each alert
- The user defines at what ETA estimates to receive a buzz for
    - Same as 5a
- The user defines which type of buzz to receive for each situation
    - Same as 5a
- The technology performs according to the settings defined by the user
    - The technology knows the specific bus the user is on as a result of the previous goals

    - The technology is constantly connected to information (from an API) about the bus's status
    - When the bus's status matches a condition set by the user, the technology fires the associated buzz
- This task is performed only once or twice since it is a setting they choose to keep for all of their routes.
- What can go wrong:
    - The user can input unideal settings, and their notifications can end up confusing or insufficient (although the defaults will always stay)

General Requirements:

- The application will be functional on both IPhone and Android devices.
- The application will require some (reasonable) amount of battery.
- The number of screens the user will need to go through will be minimal (but there will need to be a UI).
- High-accuracy GPS is preferred for optimal performance.
- Wifi- or Data-enabled (and Android or iOS).