Code Stoke Alert - Minimum Viable Product (Version 0.1) Functional Specification

Objective

The Code Stroke Alert app is a medical workflow app that enables paramedics and clinicians to communicate efficiently about a stroke case using mobile notifications and real time database updates.

This document describes the specification for version 0.1 of the application. The purpose of this version is to provide a basic feature set (notifications and CRUD data manipulation) for hospitals to install and use.

Assumptions

1. For development, the latest Code Stroke backend is running on a local development machine, using the local Flask web server (see quick start instructions on https://github.com/code-stroke/codestroke-backend).
2. Three versions of the app are required: Android, iOS, and web. The apps should all contain the same features except for the web app which does not contain the **paramedic workflow**.

Definitions

- Patient – A person receiving treatment for a potential stroke.
- Case – A container for all information about a particular patient's stroke.
- Paramedic – A user that attends to a stroke emergency and inputs the initial details about the case.
- Clinician – Any user that is not a paramedic (For example doctor, nurse, clerk).

Example Scenario

1. Paramedic arrives at location of stroke to attend to a patient.
2. Paramedic opens iOS or Android Code Stroke Alert app and logs in.
3. Paramedic chooses appropriate hospital from a list of hospitals. The list of hospitals is retrieved from a central server that hosts a JSON file containing hospital names corresponding to GPS coordinates, backend URLs.
4. Paramedic enters case details. These details are sent using "cases" API call to the backend of the hospital that was chosen previously. *(For development purposes, use locally installed Python/Flask backend, see https://github.com/code-stroke/codestroke-backend for instructions).*
5. All clinicians are sent a push notification about an incoming case. This will be sent automatically by the back end.
6. Clinicians open web, iOS, Android app by pressing the notification or by manually opening the app.
7. Clinicians view and update details about the case, via the UI of the app which sends GET/PUT requests to the hospital back end. (For v0.1, all clinicians have read/write access to all information about a case).
8. Clinicians receive push notifications when other clinicians update details about the case. The notifications that each clinician receives will depend on their "role" (as defined by the "role" field in the users table in the backend database).

Document last edited by Moe 16/6/18 4:02 pm

Note:
The mobile and web apps only need to handle incoming push notifications. They do not need to send outbound push notifications. The sending of outbound push notifications will be handled automatically by the back end.

Document last edited by Moe 16/6/18 4:02 pm

Business rules

Login, Logoff, Initial Screens

Note: Back end implementation for login and logoff is currently in progress. In the meantime, the mobile apps just require a "dummy" login screen with no back end connection.

1	If user opens app and is not logged on, display the log on screen. **For paramedics** - Log off user after a patient has been dropped off at location. (i.e. "Drop Off" button has been pressed). - Log off user if the app is force closed. **For clinicians** - Log off user after 14 days of inactivity. - Log off user if "Logout" button pressed.
2	After user logs in by entering correct username and password. **For paramedics** - Show initial patient details entry screen. **For clinicians** - Show home screen (With list of active, incoming, and completed patients).
3	If user enters incorrect username or password, show message "username or password incorrect."
4	After **paramedic** user enters initial patient details, show list of hospitals. This list comes from a central server (JSON file with coordinates). The default selection should be the closest hospital to current location (based on GPS). The hospitals should be sorted A-Z. If the central API is not available, for development purposes, create some dummy data in a local JSON file with this format: [{"name": "Austin hospital" "lat":"10034.341" "lon":"1441.453"}, {"name": "Monash hospital" "lat":"4250034.341" "lon":"425252.453"}]

Document last edited by Moe 16/6/18 4:02 pm

Notification Handler

The mobile and web apps should handle notifications being sent from the OneSignal push notification service. The notifications will be properly formatted with a title and description so there should be no extra hard coded values or strings required in the mobile or web apps.

Note from Moe: Email me for the username and password for OneSignal, which contains information and instructions on how to handle notifications for the app on Android, iOS and Web.

When an incoming push notification is handled, the mobile (or web) app should send a POST request to the back end API http://127.0.0.1:5000/acknowledgement/ with the notification id and username, to indicate that the push notification has been acknowledged.

For example if a push notification with the id "92911750-242d-4260-9e00-9d9034f139ce" is received, handled , and displayed by the mobile app and the user with user_id 25 interacts with it, the POST request to http://127.0.0.1:5000/acknowledgement/ should contain this JSON data:

{id: "92911750-242d-4260-9e00-9d9034f139ce", user_id: 25}

Note: The /acknowledgment/ API end point is still in progress and may not be implemented yet. If it is not currently available, create a mock function to mimic it.

Paramedic Workflow

1	When user presses “Next” button on Choose Hospital screen: - Send a JSON format POST request to http://127.0.0.1:5000/cases/ with the basic patient details, and status = 0 (for incoming). See schema.sql for field names. Retrieve the case_id for the newly created case from the API response.
2	When user presses “Next” button on Clinical History screen: - Send a JSON format PUT request to http://127.0.0.1:5000/case_histories/<case_id> with the inputted clinical history data. See schema.sql for field names.
3	When user presses “Drop Off” button on Editable Summary and Drop Off screen: - Send a JSON format PUT request to http://127.0.0.1:5000/case_assessments/<case_id>/ with the inputted clinical assessment data from the previous 4 clinical assessment screens. See schema.sql for field names.

Clinician Workflow

1	Home screen shows list of **incoming**, **active**, and **completed** patients. **Incoming**, **active**, and **completed** are also anchor buttons which scroll

Document last edited by Moe 16/6/18 4:02 pm

	to the selected position in the screen. Each patient name is also a button link that goes to the patient detail screen. The list of patients is retrieved using a GET request to http://127.0.0.1:5000/cases/ API.
2	When user presses a patient name on the home screen, show the patient navigation flow. The navigation flow can be navigated using menu buttons (ED, Patient Details, Clinical History, Clinical Assessment, Radiology, Management). The initially shown screen is ED.
3	The ED screen is pre-filled with data by sending a GET request from 'http://127.0.0.1:5000/case_eds/<case_id>' API A user can edit the details on the screen using the same API end point with a PUT request.
4	The Patient Details screen is pre-filled with data by sending a GET request from 'http://127.0.0.1:5000/cases/<case_id>' API. A user can edit the details on the screen using the same API end point with a PUT request.
5	The Clinical History screen is is pre-filled with data by sending a GET request from 'http://127.0.0.1:5000/case_histories/<case_id>' API. A user can edit the details on the screen using the same API end point with a PUT request.
6	The Clinical Assessment screen is pre-filled with data by sending a GET request from 'http://127.0.0.1:5000/case_assessments/<case_id>' API. A user can edit the details on the screen using the same API end point with a PUT request.
7	The Radiology screen is pre-filled with data by sending a GET request from 'http://127.0.0.1:5000/case_radiologies/<case_id>' API. A user can edit the details on the screen using the same API end point with a PUT request.
8	The Management screen is pre-filled with data by sending a GET request from 'http://127.0.0.1:5000/case_managements/<case_id>' API. A user can edit the details on the screen using the same API end point with a PUT request.

Error Handling

Show appropriate error message (toast or HUD) when an action fails.

Document last edited by Moe 16/6/18 4:02 pm