The Implementation of Historical Trend Views of Patient Questionnaire Answers on OPAL

Bejal Lewis, 260513277

I. INTRODUCTION

OPAL is a phone application designed for radiation oncology patients. It is meant to serve as a means to improve the overall patient experience, and help with data gathering, analysis and communication [2]. The application features a variety of components, which vary from tasks like checking parking availability to being able to view important lab results. OPAL was developed by HIG, the Health Informatics Group, who are a group of medical and computer science researchers at McGill University and the McGill University Health Centre, or MUHC [2]. The use of OPAL is hoped to benefit both patients and staff alike. Since OPAL is a relatively new application, the team are still working to develop new functionality. One of the current features of the application is to collect information from patients through their account, referred to as Patient Reported Outcomes. To implement this, the patients are pushed a questionnaire through their phones, and upon completion their results will be sent back to the system and are available for viewing. This project aimed to implement views in the application that would show the trends of these answers over time. In addition, values that are deemed to be unstable or dangerous should be highlighted so that they can be properly addressed. With this, it was hoped that both patients and staff will have easy and convenient access to the Patient Reported Outcomes over time, and from this be able to take appropriate action if need be.

II. BACKGROUND

OPAL is a hybrid mobile phone application that is build upon the Cordova framework, which uses AngularJS for its main functionality [2]. For the visual design components, OPAL uses the Onsen UI library. It is currently being implemented in trials at Montreal Cedars Cancer Centre. Since it is not recommended that the application connect directly with the hospital databases, Firebase is used as a storage for all data in a safe and reliable manner.

The developers for OPAL firstly researched the main reasons for patient dissatisfaction in order to pin point what OPAL should focus on. Studies have shown that patient satisfaction has a very high correlation with waiting time concerns [1][3]. Furthermore, in an RCN AOPSS survey carried out on MUHC patients, 73% of patients said that the reasons for their long waiting times were not properly explained to them, which lead to a poor review of the system [2]. Furthermore, 63% of patients expressed that they did not believe that the hospital staff did enough to accommodate them during their long hours of waiting. The results of these studies strongly indicate that research needs to be put into improving the experiences of patients. It would appear that the main concerns of hospital patients from the studies revolved around proper communication with hospital staff and administration, as well as being uninformed of the time they would be investing in treatment. With this information in mind, OPAL aims to improve patient safety, work flow, patient experience, efficiency, data gathering, and research [2].

Although still in its testing phases, OPAL has features that use machine learning techniques to estimate waiting times for oncology patients, the data for which is provided by the hospital databases and Firebase [2]. With this, it is hoped that patients will no longer have to experience the anxiety that is associated with long and unexpected waiting times. The application also contains a feature that enables a patient check-in system [2]. Since it can be tedious for a patient to need to find the appropriate desk and wait in line to

have to sign in for an appointment, OPAL makes it possible for patients to simply sign in using the application. Not only does this take stress away from the patients, but it also takes away administration burdens from hospital staff. Since OPAL aims to improve patient communication with hospital staff, patients are able to view their lab results from the application itself rather than needing to communicate directly with staff, which can come with its own delays. Furthermore, OPAL will soon have Patient Committee functionality, in which information and communication can be relayed between the patient and the MUHC Cancer Mission Patients' Committee. This committee is a way that patients can make their voice heard, so this is an important feature for OPAL. Another element that is worth mentioning is the apps way-finding system. Although currently under development, the approach is based off of NFC stickers and will provide patients indoor navigation.

The part of OPAL that will be focused on in this paper will be the patient questionnaire system. A questionnaire will help staff determine important information from a patient. For example, it might be needed for a physician to monitor how much pain a patient is in or how their fatigue levels are on a daily basis. This is where patient questionnaires are important. Physicians and staff can determine which questionnaire is most appropriate for a patient, which can then be pushed to the patients app. Upon completion, the questionnaire results are pushed back to the system where they can be viewed. Although this is useful, it would further help if one could view the trends of a patient's answers over time. This way, progress is easier to monitor and any potentially dangerous patterns can be caught early. This paper revolves around the implementation of such views in OPAL, and discusses their impact on patient experience.

III. Methods

The goal of this research project was to implement the views of questionnaire answers over time in a manner that is easy to access by both patients and staff. This section will explain the methods, in the their corresponding order, that were undergone to implement these views.

A. Observation of Previous Set up of Questionnaires

The patient questionnaire section of OPAL had already been developed before the start of this project. This section is designed as follows. Once the user has logged into the app, they are able to select the 'My Chart' button on the bottom page navigation bar.

Fig. 1. Bottom page navigation bar

From here, the user can select the 'Questionnaires' tab which will take them to the questionnaire's list page. On this page, the user can view either all of their unanswered questionnaires by selecting the 'New' tab, their partially complete questionnaires by selecting the 'In Progress' tab, or their completed questionnaires by selecting the 'Completed' tab.

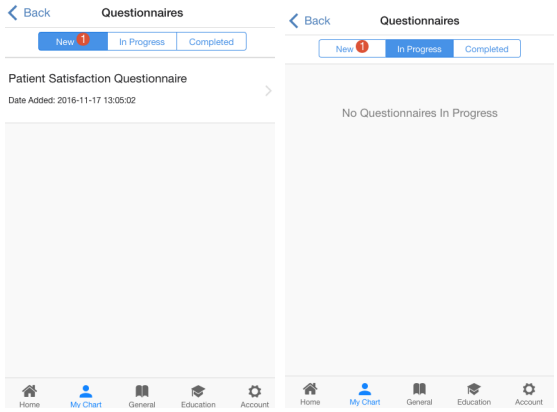

Fig. 2. Views when user selects the 'New' tab (left) and the 'In Progress' tab (right)

Inside the 'New' section, patients are able to begin answering a questionnaire, which if completed will be pushed back to the system. Similarly, within the 'In Progress' tab, patients are able to complete a said questionnaire, the results of which will be sent back to Firebase. Finally, in the 'Completed' tab, the patient can select any completed questionnaire and be navigated to a page summarizing the answers for the said questionnaire.

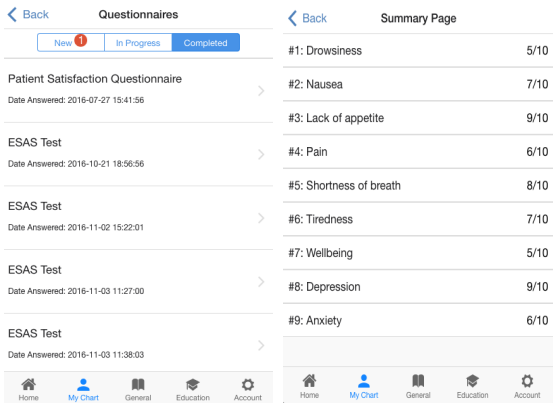

Fig. 3. View when user selects the 'Completed tab' (left) and the questionnaire summary view (right)

It is also important to understand the data structure of the questionnaire objects. All data for questionnaires, both patient answers and the questions themselves, are accessed through the questionnaires service. Through this, one can retrieve the 'Questionnaires' object, which contains all the questionnaires that are present on the patients account. One can also retrieve the 'PatientQuestionnaires' object that contains all the patient's submitted answers. These are both stored in JSON format, making it simple to parse. Using these two objects, it is possible to implement the views to show answer trends.

B. Blueprints and Drafts for Views

In order to implement the trend views in an easy and readable way, it was imperative to add to the layout of the questionnaire views instead of creating a separate page. For this reason, it was decided that it would be simplest to add another tab to the navigation bar with the title 'Trends', alongside the 'New', 'In Progress', and 'Completed' tabs. The purpose of this view would be to observe a list of all the names of completed questionnaires. The patient would then be able to select the name of the questionnaire they would like to view trends from.

Once a questionnaire name is been selected, it was then decided that the patient would be pushed to a page that would give a list of all the questions in the given questionnaire. It was not discussed that there would be a special order to these questions, thus they would be listed in the order they appeared in the questions object.

Finally, once the question had been selected, the user would be pushed to a page that had a visual representation of the answer trends. This would be a line graph if the answer was numerical, or a swipeable carousel if the answer was multiple choice, checkbox, or long answer. In the graphs, potentially dangerous trends, like a spike in pain levels, would be marked visually by a changed in color.

These ideas were drawn up and ran by team members in order to ensure that they were usable. The general consensus was that these blueprints would be appropriate for the apps current structure, and feasible to implement in the time given. One condition, however, was that the graphs needed to be similar in design to that of the lab results section of the application. For this reason, the implementation of these views closely correlated with that of the lab results views.

C. Implementation of Views

Since Cordova is built on an AngularJS framework, the above blue prints were all implemented through the creation of new controllers and views. A good summary of how AngularJS works can be found on their website. These new or updated controllers consisted of:

- **Questionnaire list controller**, which implements logic for creating the list of completed questionnaires the user can see trends for
- **Questions controller**, which implements logic for creating a list of questions from a given questionnaire that the user can see their answer trends for
- **Trends controller**, which implements logic for visually creating the trend diagrams

The new or updated views consisted of:

- **Questionnaire list view**, which shows the list of questionnaires in which the user can view trends
- **Questions view**, in which the user can see all the questions and select which they would like to view trends for
- **Trends view**, in which the user can see the graphical realization of their answers of time.

The first step was to add the new tab to the questionnaire list view. The 'Trends' tab was added through Onsen's button-bar class. In the questionnaire list controller, logic was implemented to iterate through the 'Questionnaire' object and retrieve the names of all completed questionnaires. Upon the selection of the 'Trends' tab, the list of items on the page updated to show these names. Each of these list items was then made to bind to a function in the question list controller called 'goToQuestionnaire()'. This set the relevant questionnaire object as a navigation parameter, and then pushed the user to the questions view. The navigator parameters service provides the ability to pass data from one page to another. This enabled the questionnaire object to be accessible in the questions view page. More about Onesen's navigation services can be found on their website.

Inside the questions controller, all the questions of the questionnaire object passed in the navigator were parsed. These individual question objects were then placed inside a scope parameter as an array. Using this simple logic, the questions view page could then access this array and iterate through the questions, listing out their shortened name to the viewer in the form of a Onsen list. Within this list each name had a binding to a function implemented in the controller, called 'gotoTrend()'. Much like the name suggests, this function pushed the user to the trends view, whilst adding the relevant question object as a navigation parameter.

The trends controller firstly implements logic for whether or not there is enough data to generate a trends diagram. As of now, the current minimum number of data points required is three, however this can be easily changed by future developers. If there are were not enough data points, the screen would simply explain that there was not enough data, and prompt the user to return to the questions list view. If this was not the case, the controller was then designed to access the 'PatientQuestionnaire' object through the questionnaire service, which contained all of the users submitted answers. Every question is assigned an identification number, and so the relevant question id was retrieved through the question object passed from the previous page. Using this, the PatientQuestionnaire object was then iterated through, and all answers that map to the question id were pushed to an object with their corresponding date of submission. At that point, all relevant answers and their corresponding dates, which are ordered, were able to be accessed. A method within the trends controller was created to check the type of question at hand. If the result of this method indicated that the type was numerical, then the trends page rendered to include a simple line graph using the NVD3 chart library. This line graph is passed data from the answers and dates object, and thus would be able to show a simple representation of the numerical values over time.

In order to implement logic for dangerous data trend detection, Shewhart's control chart rules were used. These rules use statistical analysis to check for 'out of control' data points [4] . The data is split up to include an upper control limit and a lower control limit, which are three standard deviations above and below the data mean respectively. These are essentially used as reference points. Using this, the rules chosen for this application were as follows:

1) If one point is outside the control limits, then a large shift has been observed and needs to be highlighted
2) If there are 6 consecutive points steadily increasing or decreasing, this needs to be highlighted

Although there are more rules that could be applied, these were the only two used for this version. These rules were implemented in the trends controller by detecting which data points match these descriptions, creating other data sets with these points, and then drawing other lines on the graph in a separate color that will naturally overlap with the original to create highlights.

If a question was not detected to have a numerical answer, the trends view disabled the graph view and instead implemented an Onsen carousel. The carousel was designed to iterate through the answers and dates object, and create a slide for each date with its corresponding answers. The user would then then able to flip through and view whichever date they please.

Regardless of whether or not the data is represented in a graph or a carousel, the full question name was retrieved from the question object and

shown clearly in an Onsen list header.

IV. Results

By creating the controllers and views mentioned above and integrating them into the current application code, the views for questionnaire trends were able to be implemented. The following describes the outcome of the methods previously described.

A. Questionnaire List View

The questionnaire's list view under the new 'Trends' tab was implemented much in the same style as the 'Completed', 'In Progress', and 'New' tabs. Since only two types of questionnaire were pushed to the test OPAL account, these are the only two seen in the list at this point. In all views, list items contain an arrow in their righter most side to prompt the user for selection.

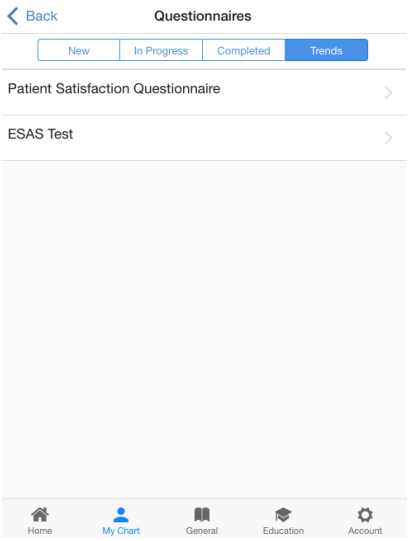

Fig. 4. Questionnaire list view under the 'Trends' tab

B. Questions View

The questions view setup wound up sharing the same design pattern as the questionnaire list view. The names are listed in the order in which they appear in the questionnaire, and each list item is linked to a function to push to the designated trend page.

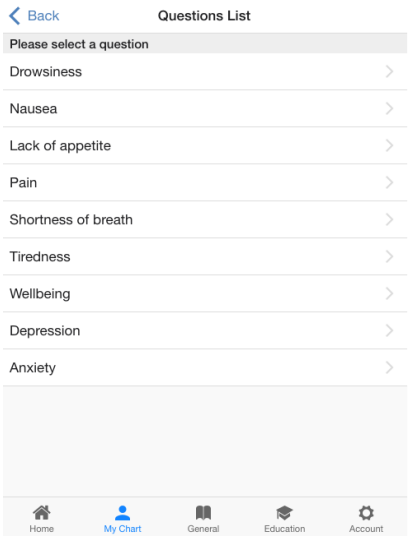

Fig. 5. The Questions view, displaying all questions for the ESAS questionnaire

C. Trends View

The graphical implementation was generated by NVD3 was made to fit the screen, whilst leaving room for a brief description underneath. The color scheme was also designed to be similar to that of the lab results, with colors being a shade of blue.

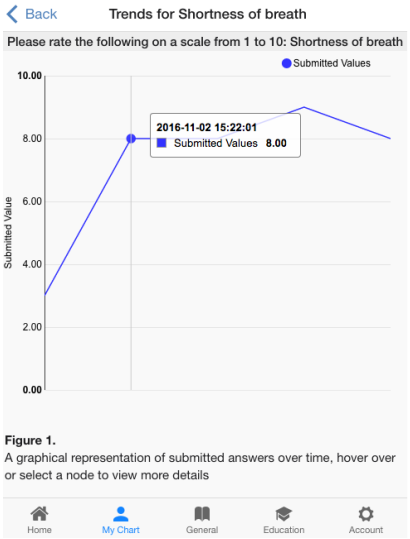

Fig. 6. Graphical interface showing numerical trends, including tool tip from hovering or clicking on node

NVD3 enabled functionality for hover or selection of graph nodes, which provided a tool tip displaying the exact submission values to the user along with the corresponding date. Since there were not enough questionnaire data present on the

phone during testing, Shewhart's rules failed and no points were marked as 'out of control'. Thus, that feature cannot be shown here.

For non numerical values, an Onsen carousel was implemented so that the user could swipe through their answers by order of date. The color scheme chosen for this was fairly neutral, with large readable text.

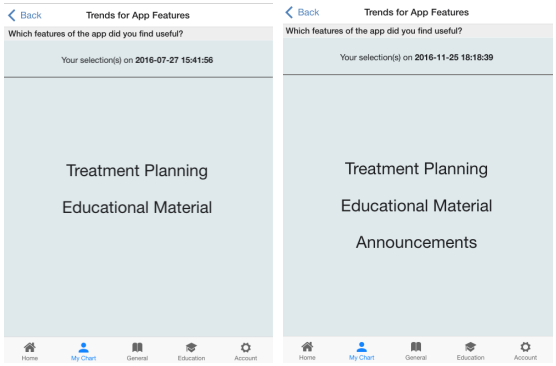

Fig. 7. Carousel items showing answers. Right slide depicts view after swiping to show next date after left slide

For when there are less than a certain threshold of data points, in this case three, an error page was also implemented. This simply explains to the user that they need to submit more answers until they are able to view trends for the question at hand.

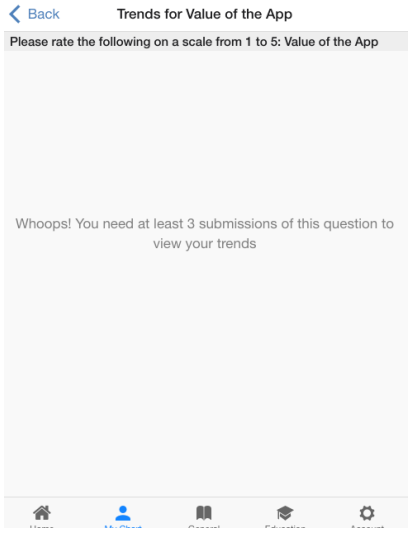

Fig. 8. Error message shown if there are too few submissions

V. Discussion

Upon presentation of the results to the team, many subjects were brought to light. These were both negative and positive issues. It was first noticed that the use of Shewhart's rules were difficult to implement below a certain threshold of data points. Most of the rules require over ten data points to be at all effective. It must be be considered that some patients may only take a few questionnaires during their time in treatment. This meant, that with the current implementation, it could potentially be impossible for patients to view the highlighted version of their numerical answer trends. In future versions, it would be better to implement Shewhart's rules under the restriction of a minimum number of data points.

It was also noticed that there were inconsistencies between the UI of the questionnaire trends and the lab results views. This was mainly pointed out to be from the fact that two different charting libraries were used to generate graphics. For the questionnaire trend views, NVD3 was used. However in the lab results page, a library called HighStock was used. The main differences in the graphs were the layout and user interaction interface. Although it would be possible to tailor both graphs to be more similar, using multiple libraries should usually be avoided. This is because the application could become bloated, and there would be no need to use different sources to achieve the same goal of creating a graph. However, it was mentioned that HighStocks are more catered to the needs of the lab result data, whilst NVD3 suits the questionnaire trend data more. It must be then decided in the future whether it is worth keeping two separate libraries on OPAL.

Another minor improvement to the current implemented trend features would be to implement a click functionality on both graph data points and inside the carousel. This could then lead the user back to a the questionnaire summary page, where all answers to the relevant questionnaire can be viewed. This was suggested simply so that the user would not have to backtrack through the pages to find a relevant questionnaire summary. It was also found that there is a minor UI glitch on the questionnaires list page, wherein switching between tabs is not seamless. There is a pause between the tab switch and the page content switch. This needs to be ironed out in future releases of the application. Another point that was made about

UI was the need for arrows on the carousel trends page. Without them, it might not be obvious to the user that they need to swipe to see results for multiple dates.

Overall, however, the design appeared to sit well with the OPAL team. The overall flow of the questionnaire trend views is easy to follow for a new user, and is quite self explanatory. Furthermore, the current implementation is such that it is easy for other developers to build upon. This is backed up by the modular nature of AngularJS. There is definitely still a lot of work to be done on the questionnaire trends section, especially in terms of fine tuning the UI as described above. Luckily, however, these advancements are very feasible, and the OPAL team aims to continue to develop this part of the application in the future.

VI. Conclusion

The overall goal of this project was to implement views to represent the trends over time of patient questionnaire answers in OPAL. This was done by creating and merging new views and controllers with the current OPAL code, building off of feedback from the team. The motive behind the project was the importance of the data submitted from the questionnaires. Physicians might be interested in, for example, a users anxiety levels after starting a certain treatment. The views implemented here would enable both the staff and patient to observe such data within a certain frame of time and act accordingly. Furthermore, when a trend is dangerous or has negative implications, it is imperative that this be noticed by hospital staff. The views implemented in this project are the first stepping stone to this being realized. Future work includes fine tuning the current UI, and implementing more of Shewhart's rules to detect more types of patterns in the reported trends.

References

[1] Fitch M.I., Gray R.E., Psycho-Oncology 2003; 12: 664674

[2] Hendren, Hijal and Kildea, Wait times in radiation oncology: Addressing the pain of waiting, MUHC Oncology Grand Rounds, March 2nd, 2015

[3] Paul C., Carey M, et al, Eur. J Cancer Care 2012; 21, 321329

[4] Tague, Nancy R. The Quality Toolbox. 2nd ed. N.p.: ASQ Quality, 2006. 55-158