Feature: Alerts

Feature Developer: Joshua Gherman
Date Developer submitted: 3/23/2023

Reviewer: Frank Curry
Date Review Completed: 3/28/2023

Summary

Major Positives

1. The front-end designs are well thought out and have a natural flow to them.
    - With the exception of the admin view, the front end views are easy to understand and how to reach them from the previous page. They also contain simple and readable user interfaces.
2. Considering options that the writers of the Business Requirements Document may not have.
    - One of the failure cases was not one that we had specifically thought of, but identified a clearer way of identifying a failure case.
3. The inclusion of checking the JWT.
    - Although this was still being explored when developing the diagrams they were implemented knowing that we are approaching JWTs for authorization.

Major Negatives

1. The automatic looping in the design.
    - The design for automatically reloading without the consent of the user is a poor design choice. This can lead to a loop if the feature is down or never fulfills a condition where it can load. We were told in the past that automatic reloading should not happen and instead have the user make the request again.
2. There were some designs for user cases that were missing.
    - There was a service user story that was missing in the diagrams where a service user would receive an alert. There were also missing failure cases where a user never receives a notification they were meant to receive and one for the list of notifications not loading.
3. The diagrams did not accurately show what was happening in the data store layer.
    - On multiple diagrams the Sql statement would just be “SELECT * FROM dbo.Alerts” which would return the whole table. This was clearly not the best way to design this and likely was not the designers intentions.

Unmet Requirements

1. Valid Alert Input Verification.
    - The designs did not take into account any of the verification required by the BRD when it comes to what is in the alert.
2. Any kind of design for service users.
    - The Service users have a functional requirement that they would receive alerts that may not be available to a Regular user.
3. A scenario where the notification list exceeds 100 and the oldest notification is marked as read.
    - Based upon the Alert functional requirement 8, the system must mark the oldest notification as read.
4. Having marked as read notifications expire after 7 days.
    - Based upon the Alert non-functional requirement 2, the system must delete

    notifications if they are marked as read after 7 days.

5. A warning message to user informing them of their full notification list.
    - Based upon the Alert non-functional requirement 9, the system must inform the user that their notification list is full when at or exceeding 100 notifications.

Design Recommendations

- The automatic looping in the Low-Level Designs need to be replaced with something that has stopping points. Having the automatic looping will result in unnecessary automation that could be hazardous to performance as it may never stop. Adding an option for the user to reload the notifications to replace the point where it is automatically called again should be considered. It's a positive that it is less intensive on the system, just the same call but less frequent. The negative is that you have to make a button for the user to request again.
- The username or user ID should be the main way to identify alerts and not user hash. While not explicitly shown in design, it will allow quicker fetching of alerts and the data will be deleted when a user deletes their account. The positive is that it increases performance compared to user hash. The negative is that when a user deletes their account their alerts may go with it so an additional delete statement is required.
- Make sure to do any searching for something in the datastore, on the datastore. The design was not exactly clear how it was getting the alerts that belonged to the user, making sure to leverage the relational database language to filter data. The positive thing is you are not dragging an entire table into the backend and less intensive on the CPU. The negative is it may be slower to fetch specific items.
- For the datareader, possibly use the async version. The async version can possibly increase performance based on Remus Rusanu's experience on stack overflow (https://stackoverflow.com/questions/9432647/any-disadvantage-of-using-executereader async-from-c-sharp-asyncctp). The positive, possible increase to perform and can use additional queries on the database. The negative, I do not have full verification if it does increase performance and additional async precautions in the back-end.
- For the front-end, the scope for the pin types should be narrowed to check boxes. Alternatively you can leave it as a text field with verification. The positive aspect is users can only select from the curated pin types, leaving little room for user error. The negative is in the future if new pin types are added then the page would need to be updated. An update would not really be a problem if left in a text field format except for the verification of text would need to be updated.

Design Recommendations

- For the zip code user experience, I suggest that through a front-end test and end-to-end testing that different options are selected. This is in relation to the user being able to select specific pins to receive notifications from. Testing to see if it saves them correctly and filters out the ones that are not being accepted by users.
- For the alert setting, test to make sure that they have it enabled. This can be done through a unit by making the setting not go to the back-end by failing. An end-to-end test can be done as well. This is because of the non-functional requirement 1 for Alert where all functions are off by default meaning you have to test if they are enabled.

- For the datareader, it would be best to make an integration test where you grab data and put it into variables if you were going to go that route. If not, then simply retrieving them async in under 2 seconds should be sufficient.
- For the username as a column, I do not think there needs to be a test for a change in the data table design.
- For the automatic looping fix, test the front-end in an end-to-end test where it fails to load the notification list properly the first time. Then press the reload button to request again, having it properly load this time. This is to fulfill the failure operations where the notifications either do not load, have incorrect notifications shown, or never receive the notification.

Sections

1. Front-end
2. Alert Successes
3. Alert Failures

The sections are specific for each aspect for the design that was presented. This is a more detailed explanation and breakdown of how I thought the designs were presented. Each section has a major positive, major negative, minor positive, minor negative, novel design recommendation, superficial design recommendation, and test recommendations. The superficial design recommendations are not a major contribution to the design and are merely my thoughts on how to design certain elements based off of Josh's design. The superficial designs do not need to be followed.

Major Positives

Section	Positive Attribute
1	With the exception of the Admin Alerts view, I can see a clear flow of how each page leads into the other without any kind of arrows or indications. I believe that this shows the alerts webpages are well designed when it comes to a regular user experience. For the “Notifications” view I think that having the “Notification n” is a great conveyance for the developer of the front-end. It shows that they could potentially go on until they reach a certain value. However, we have already stated that we should hold up to 100, but this could be extended based on the business requirements so it shows extensibility. For the “Notifications” view the addition of the “mark all read” button is great. The business rules do not require it but I think that it is a great addition for those that receive a lot of notifications they do not want to read. Alternatively, for returning users that do not want notifications that are no longer relevant it saves time for the user.
2	For success, the inclusion of JWT is good. Although this was still being explored when developing the diagrams they were implemented knowing that we are approaching JWTs for authorization. The pin alert out is a bit subtle but it shows consideration into how the users are going to receive the alerts, something that I did not even consider until late into the review.
3	The addition of receiving a notification that was meant for another user is good because it is something that we did not initially consider when writing the BRD. It is a more specific version of “the list of notifications does not show the correct notifications.”

Minor Positives

Section	Positive Attribute
1	For the Alerts notification on the home page view the small red dot signifying that they have new or unread messages is good. It is not intrusive to the user experience to have it but still catches the eye with a contrasting red color.
2	The simplicity is good when it comes to understanding the overall picture of the success diagrams.

3	While on a misleading journey for designing the “automatic reloading,” I think that the dedication to representing the flow of the functionality can help those who are reading through it the first time.

Major Negatives

Section	**Negative Attribute**
1	For the Alert settings having the “Accepted Pin Types” be blank does not give me an indication what kind of options it should have. The input type of the setting is not told through this image. Is it a text field for input where each pin type should be typed out? If so, as the user, how are they supposed to know the pin types? (see Novel Design Recommendations 1.1) For Admin Alerts from the pdf it does not show how the admin would be able to reach that view, maybe it is shown on the figma diagrams but that is not what is shown so I can only comment on what I see. If additional views were shown from the point of an admin it would be sufficient.
2	For the user there is a precondition that the user must fulfill in order to have the notification sent. The precondition is that they need to have the “Send Notification on Pin” setting turned on in the settings. The application should not have those on by default as the first Non-Functional Requirement for Alerts in the Business Requirements Documents states that “All alert functions, with the exception of Admin alerts, are off by default.” (See Novel Design Recommendations 2.1) For AlertS1aa there does not appear to be anything done in the data store layer, once it reaches that point it does not do anything. I assume that it does the sqlStatement “SELECT * FROM dbo.Alerts” in this case we are not looking into the database for specific alerts for a user we are grabbing all of them. With no commands that is a large amount of data could be brought into the backend and from there will there be a search? That search for the user that we are looking for after bringing in the data might be too much for the cpu to handle. (See Novel Design Recommendations 2.2)
3	For the Alerts F2b there is a big issue. When displaying the error and then having an automatic response to try and reload notifications, please do not do this. There is a big issue with a design like this, what would happen if the automatic response fails? It would likely be stuck in a loop where it will continue to attempt to connect with no success. Having the diagram this way does not fully show a design for an automatic reloading of an element. There would need to be some kind of loop back to the automatic response from the user to main.js. However, the best way to do this is not to have it be automatic. (See Novel Design Recommendation 3.1) I think that you are missing a few diagrams demonstrating some of the failure scenarios described in the Business Requirements Document (BRD). From the BRD you did not show “The list of notifications does not load” and “The user never receives the notification that they were meant to receive.” I understand that it might be hard to demonstrate a scenario where the user

	did not receive a notification because it should without fail reach them and how are we supposed to verify if it did reach them is something of a user responsibility to identify. Also from the BRD I can see how the automatic reloading of notifications came into the Low-Level Diagrams with the “Reloading” error message. While I believe that the text put into the BRD must be followed to the letter, in this case I forgot to change the system message for this one and possibly others. However, I believe that you should follow the system messages in the BRD to the letter meaning that you would be missing all the designs that have unique error messages. (See Novel Design Recommendations 3.2)

Minor Negatives

Section	Negative Attribute
1	For the Alert Admin Alerts the “Zip Code Range” is difficult to understand. Zip Codes in California do not necessarily enumerate linearly according to area. For example if you look at the zip code for CSULB (90840) and the neighboring zip code (90815) they are not adjacent numbers. Therefore it may not necessarily be an easy task to just use a range as they could be in areas that may not need an alert. The same could be said for the zip code range in the settings except for that one it might be easier to allow users to type in the zip codes they wish to receive notifications from rather than having them upload a csv. (https://gis.data.ca.gov/datasets/CDEGIS::california-zip-codes/explore?location=33.789273%2C-118.105464%2C13.98) (see Superficial Design Recommendations 1.1) For the Alert settings having the “Receive Event Notifications” and “Send Notifications on Pins” be on or off is good, but having them be green and red as default does not tell me whether they are enabled or disabled. Would a user be able to tell if they have the “Receive Event Notifications” active if they just created an account? I would not from the design shown. (see Superficial Design Recommendations 1.2)
2	In the diagram from the data access layer going to the data access layer (SqlDAO -> SqlDAO), “var alerts” and “string sqlStatement” are declared on the same line the issue is that I did not immediately recognize that it was even supposed to be two separate statements. This issue is for most of the other diagrams as you likely copied one template that had the same issue. For the “Alert Manager” layer, the “if User.this == true” statement has multiple expected parameters. I highly doubt that these parameters are all the same for different variables. (See Superficial Design Recommendation 2.1) Something that is missing from the diagrams is the return value when making a function call. The function calls should be something like “somFunc(param:*String*):*int.*” You did apply this function call in some of the diagrams like in “NotificationController” where you showed the return type. (See Superficial Design Recommendation 2.2)

3	I understand that the diagram was copy and pasted for the sake of brevity when it comes to most diagrams as the structure between success and failure have little difference. However, the failure still lacks what is being done in the data store layer. For the AlertsF1a at the "Notification Controller" there must be some kind of conditional statement to pass along the axios response that the list of Json(Alerts) was empty. As it stands in the diagram I assume that there is some kind of conditional in the controller that is not shown. It would be best to show it to provide a thorough design.

Unmet Requirements

Section	Unmet Requirement
All	**Valid Alert verification:** There is not anywhere in the design where the verification of the alert is checked. I suggest that you find somewhere to verify the characters used in each alert. Most likely in the front-end before the user has a chance to send it to the back-end. **Any kind of design for service users:** While the service users do not exist yet they do have a functional requirement from the BRD. "The system must have service users receive notifications from user requests" while it is unknown how the service users will be implemented right now having some form of design is important. When it is developed you might not have to change a whole lot from the original design and it will be easy to identify where it needs to be changed. **A scenario where the notification list exceeds 100 and the oldest notification is marked as read:** Based upon the Alert functional requirement 8, the system must mark the oldest notification as read. This is not explicitly shown within the diagrams presented. While the limit for the list is set at 100 when retrieving it, notification 101 or greater should be marked as read. **Having marked as read notifications expire after 7 days:** Based upon the Alert non-functional requirement 2, the system must delete notifications if they are marked as read after 7 days. **A warning message to user informing them of their full notification list:** Based upon the Alert non-functional requirement 9, the system must inform the user that their notification list is full when at or exceeding 100 notifications.

Novel Design Recommendations

Section	Recommendation

1.1	For correcting this mistake you should probably have this be a field of check boxes for the HTML. Using check boxes allows the user to know the options that they have available and choosing from the curated options. Since we have control over the options for the check boxes there would not necessarily be a need to verify any of the information that we receive since we limit their options. If you wish to continue with a text field input you would need to write the options that they should be allowed to input and then verify that they wrote them correctly in the front-end and back-end. You should view this link if you want to see a demonstration of check boxes for HTML. (https://www.w3schools.com/tags/att_input_type_checkbox.asp) **Positive:** Users can understand what pins they are accepting or denying. **Negative:** In the case that more pin types are added in the future a checkbox area would need to be expanded to show those new pin types. In the case of a text field, validation will be needed to see if the pin types entered are valid pin types. If more pin types are added than the text verification is in need of an update as well.
2.1	For the lack of precondition for the user when they have any form of notifications enabled I suggest putting a precondition before the user interacts with the application. Having this precondition would inform those who are creating the application to know that they need to check and see if they have the setting enabled. This precondition for the setting will need to be checked somewhere in the front-end or back-end for when they are placing a pin. **Positive**: Developers can understand that they need to have the setting enabled. Validation for the setting prevents errors. **Negative**: For the diagram, it is for developers only adjustment.
2.2	Assuming that this is the way that it was intended I would recommend searching for the user related alerts in the database rather than on the system running the back-end. The system could suffer from a slow down of looking at the possibility of 10000 entries to parse for one user. Try and leverage the relational database language to get what is needed quickly. After reviewing the diagrams once again I see that you are using sqlcommand.executereader. I still recommend searching for the user on the database side but you can return a list of alerts by assigning them to variables shown in this website (https://www.c-sharpcorner.com/blogs/executereader-executenonquery-and-executescalar-in-adonet). Also consider using the async version of executereader some details from this stackoverflow.(https://stackoverflow.com/questions/9432647/any-disadvantage-of-using-executereaderasync-from-c-sharp-asyncctp) While I cannot verify the integrity of the answer it is something to think about since the option is available. **Positive**: You are able to utilize the relational database to get the data that is needed to be checked. **Negative**: Having async could have unknown consequences as of this moment on the datastore and program flow.
2.3	For the database it should probably use just the username for the account (i.e email) to keep track of alerts belonging to a specific user. This is because the alerts should be fetched fairly quickly and searching the database will be quick. The alerts do not need to be kept after

	someone needs to delete their account for any reason. Therefore, for performance, it would be good to use just the email. **Positive**: Increased performance in searching for a specific user on the datastore. Deletion of unneeded data when a user deletes their account. **Negative**: Additional calls to the database when deleting the user account.
2.4	Something missing from the design is zip code validation. There is a limited range of zip codes 90001 to 96162. There should be some sort of check to see if the inputted zip code is valid. **Positive**: System can accept inputs without a user error. **Negative**: Additional checks in the code with almost non-existent performance impact.
3.1	The main issue with the automatic reloading of elements or features in a program is that they can get caught in a loop that can waste resources in addition to other things. The low-level diagram did not show this loop. A better way to design this kind of reloading is to have it be manual on the part of the user. Setting a button of some kind to allow the user to reload the notifications, sending the request to get them again, would be an improvement. This would also reduce the size of the diagram. There was a reason that we changed the Business Requirements Documents to not include anymore “automatic” reloading or refreshing, this is because at some point the instructor had told of these dangers. **Positive:** Avoid any kind of looping automatically done by the system that could cost performance. **Negative**: It must rest on the user's shoulders to request the notifications again.
3.2	I recommend that you follow the system message displaying a certain text from the BRD. Following the BRD will only benefit us because we followed what we said we were going to do. In the case of the system message saying “Reloading” that is more of a typo that we left in after revising multiple times and that portion of the text could potentially be ignored. I say that it should be ignored in this case because it could be misleading for the user to read this message then give them an option to reload. Consult the client for proper steps on how to deal with this case. **Positive**: The message is more accurate to the BRD in regards to all relevant system messages to the users. In the case of an outdated message, it is more accurate to the users based on what is actually available to them. **Negative**: Potentially going against what we have said we would complete and what the client agreed to.

Superficial Design Recommendations

Section	Recommendation
1.1	A possible way to select zip codes could be through uploading a csv file with the zip codes that would contain all of the places that need to be alerted. It could be a combination of both where the admin might be able to have an alert for a singular zip code input in a text field and a csv file upload for a bulk alert to multiple zip codes.

	Positive: Admin users would not need to input each zip code individual, within the application itself. **Negative**: This does not really improve the speed of inputting the valid zip codes. The admin user would still have to type out each valid zip code. In addition, you would need to add extra functionality for something for the sake of comfort.
1.2	A possible solution to the on/off buttons would be to gray out the selection that is not active (https://www.w3schools.com/cssref/tryit.php?filename=trycss3_js_filter). Another possibility would be to make it a HTML radio selection instead, that way the user would be easily able to see which option is currently active (https://www.w3schools.com/tags/att_input_type_radio.asp). **Positive**: Users will know what they are selecting and what they already have selected, improving user experience. **Negative**: Additional work for a cosmetic change, making it low impact.
1.3	For the Alert settings I would consider putting a save button to save the current settings of the webpage or updating their settings when they leave the page. Either is a viable option. This is because as a user they want to know that their options were saved particularly if they did not want to send notifications. **Positive**: Users will know that they are saving their settings so it is not entirely based on assumption. **Negative**: Additional button functionality, potentially, or a small area of text informing the user.
1.4	For the "Notifications" view having the "# Notifications" part below the title, I assume that this is the count for the total number of notifications. If this is the case then a simple "of" may clarify. So, it would be "# of Notifications" making it more readable to someone that was developing the front-end. If this is not the case then that portion is poorly demonstrating what it is supposed to be. **Positive:** increased readability for developers and programmers. **Negative:** increased file size (bytes), for developer readability only.
2.1	For the variables or methods in the Alert model I would recommend making the parameters have different names would benefit the people who have to implement the design. In this case it is yourself, but in the future it could be someone else, having different parameter names can differentiate what should be there. For example "alert.zip(expected)" could be "alert.zip(expectedZip)." It is something minor that would likely change during the implementation while working on it but it could be beneficial in future projects. **Positive**: Easier for a developer to differentiate parameters. **Negative**: It has no impact on the system, it is for developer readability only.
2.2	I would recommend having it in the format stated earlier, "somFunc(param:*String*):*int.*" The format helps the implementer understand the return types and parameter types. Although it is already shown what the return type is when the function completes with the return arrow the redundancy reinforces what it should be. **Positive**: Easier for a developer to understand how to program functions with just the diagram.

	Negative: Change has no impact on the system, it is for developer readability only.

Test Recommendations

Section	Recommendation
1	For the zip code user experience, I suggest that through a front-end test and end-to-end testing that different options are selected. This is in relation to the user being able to select specific pins to receive notifications from. Testing to see if it saves them correctly and filters out the ones that are not being accepted by users.
2	For the alert setting, test to make sure that they have it enabled. This can be done through a unit by making the setting not go to the back-end by failing. An end-to-end test can be done as well. This is because of the non-functional requirement 1 for Alert where all functions are off by default meaning you have to test if they are enabled. For the datareader, it would be best to make an integration test where you grab data and put it into variables if you were going to go that route. If not, then simply retrieving them async in under 2 seconds should be sufficient. This is in relation to the non-functional requirement where the list must display the notifications in less than 2 seconds. This can be done end-to-end as well. For the username as a column, I do not think there needs to be a test for a change in the data table design.
3	For the automatic looping fix, test the front-end in an end-to-end test where it fails to load the notification list properly the first time. Then press the reload button to request again, having it properly load this time. This is to fulfill the failure operations where the notifications either do not load, have incorrect notifications shown, or never receive the notification.