Space Surfers: Reservation Confirmation Email

Developer: Sarah Santos

Developer Submitted: 03/15/2024

Peer Reviewer: Sarah Phan

Review Completed: 03/17/2024

Major Positives

- Provides a comprehensive diagram by detailing the complete sequence of steps preceding the confirmation email being sent.
    - This helps to give a better understanding of the flow of logic for this service as it gives the necessary context required for the service to be called on.
- Details the flow of logic for all steps within the reservation confirmation email service for data access as well.
    - By showing how a configuration file is required to connect to the data store, this clarifies that a configuration file is a dependency necessary for this service to function properly.
- Defines all objects needed for service to work
    - Demonstrates a close attention to detail by including all instances of classes that are made. This makes it clear that the reservation confirmation email service will require CustomSqlCommandBuilder, SqlDAO, Response, ConfigService as dependencies.

Major Negatives

- No clear separation of concerns in regards to sending a confirmation email and adding to the calendar for that space reservation.
    - The email service should only be responsible for sending emails. Because of that, it may be helpful to create a class for the reservation confirmation email service that will be called on when a reservation is made. This class will likely have the method to send the email with the given arguments, such as

reservationInfo. The createCalendar method could be a part of this service or may be its own class. If it is its own class, it might be called somewhere in this reservation confirmation email service.

    - Another option is to create a method within the MailSender service, which will also be used for other features. Since we will be sending a variety of emails throughout our application, it may be useful to have separate methods for each type of email within this MailSender service.

- Can provide a more low level breakdown of the request handling process
    - In line 3 of the diagram, we see an AJAX post being sent from the controller to the service. While this shows the interaction between the frontend and the backend, it is more high-level. Instead, it would be more granular if we showed how there is an HTTP POST specifying what is posted from the UI to the controller. Then, the information would be sent from the controller to the service. After the service does its part, an HTTP response 200 (or 400 for errors) would be sent back to the UI.

Unmet Requirements

- Data store connectivity error handling sequence
    - If attempting to access the data store fails, we would receive response.HasError = false. This would be returned to the confirmation email service, resulting in the following message displayed to the user as per the BRD: “Database is not able to process retrievals right now, please come back later.”
- Errors for sending an email
    - The errors that can occur from attempting to send an email include error handling for SMTP command errors, SMTP protocol errors, authentication errors, errors when our end isn't connected to the SMTP server, and a general exception error. The response object should be made before sending an email since if an error occurs, response.HasError would be false. This will allow us to check for it and display a message to the user accordingly.
- Need to address the scenario where a user is prompted to request another confirmation email
    - As per the BRD, if an issue persists where a user does not receive their email, they are displayed the message: “Error generating email. Please request another confirmation email.” Because of this, we need to provide the user with an option to resend an email. In this case, the diagram would begin from when the user makes this request instead of when the user makes the reservation.

Design Recommendations

Resend email option

- As mentioned before in Unmet Requirements, an option to resend the confirmation email should be available to the user. In the diagram for this scenario, it would not be necessary to show the sequences from making a reservation leading up to the confirmation email being sent. However, since information, such as space ID and reservation time, is needed it may be useful to put this resend option somewhere when the user goes to view their reservation for this space after making the reservation.
- Pros
    - Allows user to resend confirmation email in cases where they do not receive the email
    - Useful in cases where errors occur within the backend, such as from the SMTP client
    - Improves user experience through flexibility and convenience
- Cons
    - Potential for users to abuse this option. This may result in flooding the SMTP client with a large volume of requests
    - Requires additional development efforts which may not align with the estimated hours for our project

Recommendations for line (6) in the diagrams

- For the sendConfirmation method from the emailService, the types for reservationInfo and calendarLink should be specified. Additionally, it might be useful to have the information regarding the reservation be put into a custom object since it allows for better encapsulation, reusability, and maintainability of code. For failure scenarios regarding email errors, we can also check them around line (6). Since we send the

email here, if an error were to occur, we would return the controller would receive response.HasError = true and would, in turn, return an HTTP 400 response back to the UI. From there, we could inform the user that the email failed to send.

- Pros
    - Clarity on parameter types
    - Encapsulation will promote cleaner and more modular code since all necessary reservation information will be grouped together
    - Reusability from having a custom object for all reservation information since other services, such as waitlist, will also require this information.
    - Ease of maintenance from making a custom object since possible future changes to what belongs in reservation information can be updated in the object.
- Cons
    - Requires more time to implement the custom object. This may not align with the estimated hours for our project in regards to this component.

Test Recommendations

Unit Testing

- Test the possible errors when sending a confirmation email. Errors from sending an email include SMTP command errors, SMTP protocol errors, authentication errors, errors when our end is not connected to the SMTP server, and a general exception error.
    - Authentication errors occur from having incorrect credentials for SpaceSurfer's email account. Since we use the app password for this account, testing for this error might be done by incorrectly configuring the email or the app password.
    - As for SMTP command and protocol errors, since these originate from the MailKit client itself and how it interacts with the SMTP server, it may be harder to test. However, creating mock servers and commands to that server may be an option.
- Creating a mock of the email service to verify the correct email content (subject and body)

E2E Testing

- Simulate user actions to ensure email is received within 3 seconds.
- Ensure that no email is sent if there are errors when making the reservation, such as invalid or missing inputs
- Ensure that correct errors are displayed to the user as per the BRD
    - For cases where operation timed out: "Time Limit for Operation Exceeded."
    - For cases where email generation fails: "Error generating email. Please request another confirmation email."

- For cases where the database is offline: “Database is not able to process retrievals right now, please come back later.”