# **Defining the expected result in detail**

Imagine, a developer has finished a newsletter subscription module that technically works as expected. Meaning, if you try to sign up with something like “erhfejgh” it does not put it into the database, when you sign up with a proper email - it adds it to the DB, and if you sign up with the same email again, it does not create a duplicate.

However, the initial ticket did not feature information about the notifications to show to the user and the developer was not proactive enough to think about the UI and missed adding notifications to explain to the user what was happening.

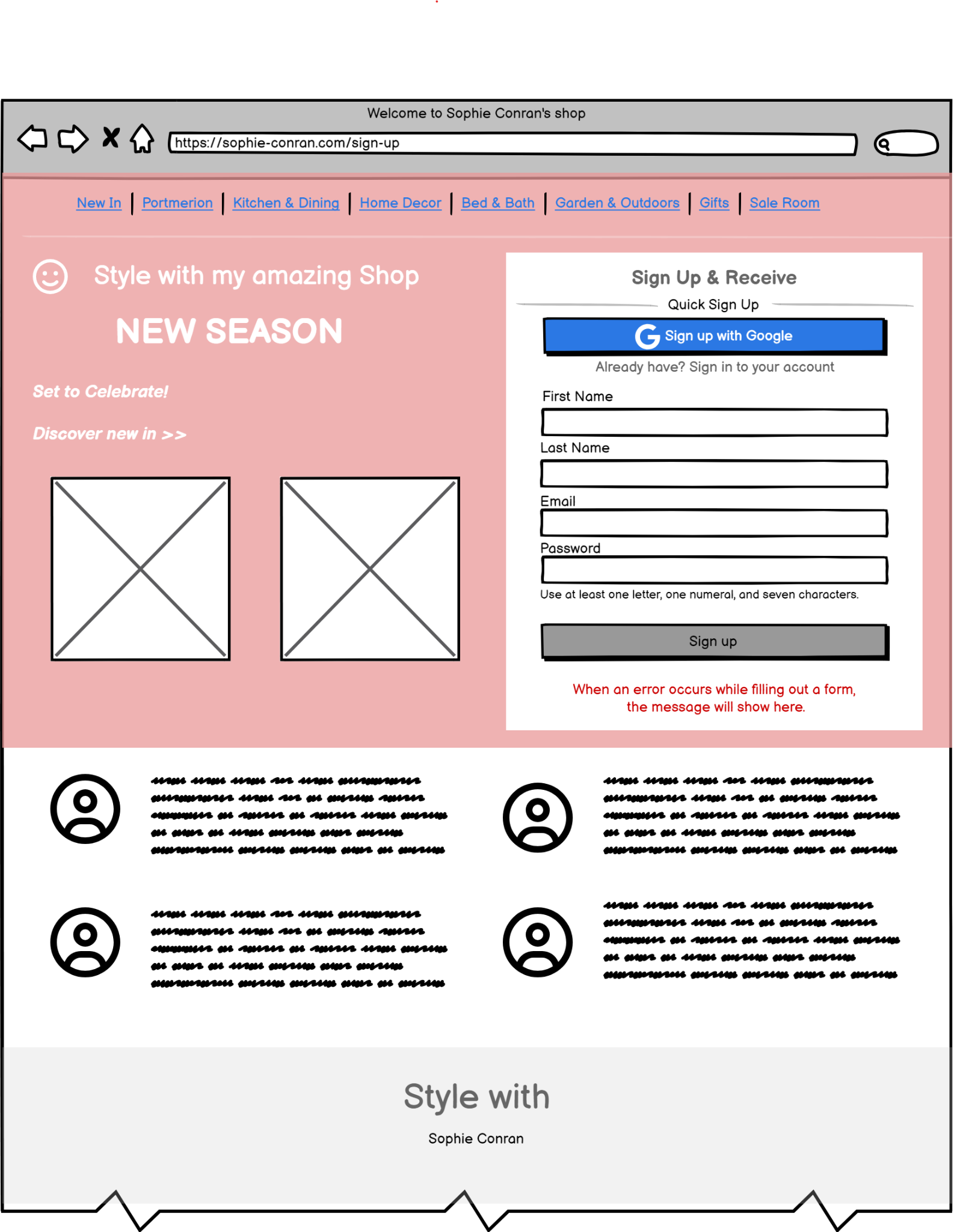
So, now whatever the user is doing, the feature behaves correctly, but it is “silent”, It does not tell about the error if the email submitted is not in the correct format and it does not tell “thank you”...

**Your task & expected result**

Please, create a task for a developer to add specific notifications on specific actions. Visually, these should appear right under the newsletter subscription box, there is plenty of space, so no need to think about overlays or toast-notifications

**Subscription for Adding User Notifications to Newsletter**

The newsletter subscription has been successfully developed with proper functionality, allowing users to sign up using valid emails and avoiding duplication. However, it lacks user notifications to provide feedback on the actions performed by the user. The objective of user experience is to add notifications for specific actions in the newsletter subscription module. We need access to the codebase of the newsletter subscription module and any relevant UI assets for displaying notifications.



[**A good task feature**](https://www.notion.so/5c150993d2ed4b6ba72d8c6c6b9cdca4?pvs=21)**s**

**Title - Invalid email format notification:**

**Why? -** Email validation is a crucial step in maintaining accurate and reliable data within a business database. When an incorrect email address is not included in the database, it helps to ensure data integrity and prevent potential issues that could arise from inaccuracies in customer information. It’s important for Communication Efficiency (Inaccurate email addresses can lead to failed communication attempts, including undelivered messages, bounce-backs, and spam reports. This can harm customer relationships and communication channels. ) Accurate customer information is crucial for personalized marketing, customer service, and business analytics, Reputation Management and Cost Efficiency too.

**ToDo -** When a user submits an email with an invalid format (e.g., "erhfejgh" instead of a proper email), display a notification under the subscription input box.

**Content -** The notification should clearly state that the submitted email is not in a valid format. Use a visually distinguishable color (e.g., red) for the notification text. For example: “The submitted email is not in a valid format, please retry again!”

**Visual/Wireframes**

****

**How to test? -**

* **Test scenario: Testing Invalid Email Address Validation**

**Test data:** Input: "erhfejgh"

Expected Result: The system should not allow the invalid email address "erhfejgh" to be added to the database.

**Following steps:**

* Access Sign-Up Form: Navigate to the sign-up form on the website.
* Enter Invalid Email Address: In the email address field, enter the invalid email address "erhfejgh".
* Submit Form: Confirm with the "Sign up" button to attempt to sign up with the invalid email address.
* Validation Check: The system should validate the email address in real-time and display an error message next to the email field, indicating that the entered email address is not valid.
* Expected Result:

The system should prevent the user from submitting the form with the invalid email address "erhfejgh".

The user should see an error message (e.g., red, for the notification text) informing them that the email address is not in the correct format.

* Database Check: check the database to ensure that the invalid email address "erhfejgh" was not added to it.

**Title - Successful adds into DB with a Proper Email:**

**Why? -** In essence, adding valid email addresses to the database is the foundation for building and maintaining strong customer relationships, effective marketing strategies, and data-driven decision-making. It's a fundamental aspect of modern business operations in the digital age.

**ToDo -** UI Notification: Upon successful sign-up, the UI should display a clear and positive notification indicating that the sign-up was successful, under the subscription input boxes.

**Content -** The notification might include a message such as "Congratulations! You've successfully signed up with the email address [user@domain.co](mailto:user@domain.com)m." Use a visually pleasing color (e.g., green) for the notification text.

**Visual/Wireframes**

****

**How to test? -**

* **Test Scenario: Successful sign-in with Proper Email:**

**Test data:** Input: "[user@domain.com](mailto:user@domain.com)"

The system should add the valid email address "user@domain.com" to the database. The UI should provide clear notifications of the successful sign-up.

**Following steps:**

* Access Sign-Up Form: Navigate to the sign-up form on the website.
* Enter valid Email Address: Test the system with a valid email address (e.g., "user@domain.com") to ensure that it is successfully added to the database.
* Submit Form: Confirm with the "Sign up" button to attempt to sign up with the valid email address.
* Typographical Errors: Test the system with email addresses containing typographical errors (e.g., "user@domain..com") to verify that these are detected as invalid.
* Domain Validation: Test the system with email addresses containing valid syntax but an invalid domain (e.g., "user@fakedomain.com") to ensure such addresses are rejected.
* Database Check: You must check the database to confirm that the valid email address "user@domain.com" was successfully added to it.
* Expected Result:

The system adds the valid email address "user@domain.com" to the database.

The UI provides a positive and noticeable notification confirming the successful sign-up.

**Title - Signing up with existing email without duplication:**

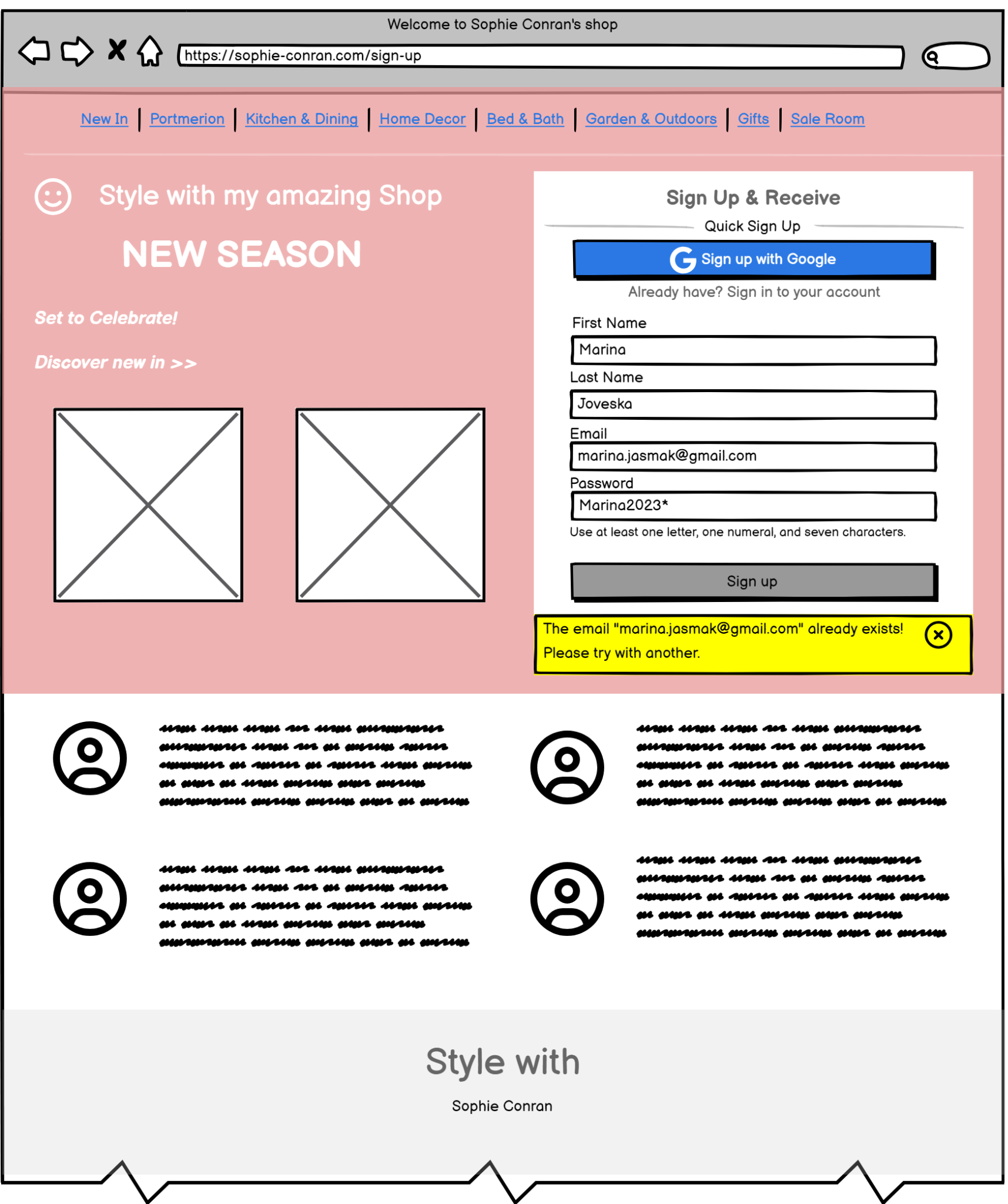
**Why? -** Preventing the creation of duplicate entries in the database for the same email address is essential for maintaining data integrity, improving operational efficiency, and providing a better user experience. Creating duplicate entries can lead to confusion, inaccurate analytics, and potentially annoy customers with redundant communications.

**ToDo -** Existing Address: Test with an email address that already exists in the database to confirm that duplicates are not allowed.

Edge Cases: Test with extremely long email addresses, special characters, and various valid formats to ensure robust validation.

**Content -** If a user attempts to subscribe with an email that already exists in the database, display a notification under the subscription input box. The notification should inform the user that the email is already subscribed to and provide guidance on managing their subscription preferences. For example: “The email "[user@domain.com](mailto:user@domain.com)" already exists, please try with another”. Use a distinct color (e.g., yellow) for the notification text.

**Visual/Wireframes**

****

**How to test? -**

* **Test Scenario: Preventing Duplicate Email Entries:**

**Test data:** Existing Email in Database: "[user@domain.com](mailto:user@domain.com)"

**Following steps:**

* Access Sign-Up Form: Navigate to the sign-up form on the website.
* Enter Existing Email Address: In the email address field, enter the existing email address "user@domain.com".
* Submit Form: Click the "Submit" button to attempt to sign up with the existing email address.
* Validation Check: The system should validate the email address and display an error message next to the email field, indicating that the entered email address already exists.
* Database Check and Verification: You must verify that the existing email address "user@domain.com" is not duplicated in the database and check.
* Expected Result:

The system prevents the user from submitting the form with the existing email address "user@domain.com".

The user should see an error message informing them that the email address is already in use.

**Notifications UI setup:**

1. **Notification Display Duration:** Ensure that each notification remains visible for a reasonable duration (e.g., 5 seconds) before automatically fading out. Implement a smooth fade-in and fade-out animation for the notifications.
2. **Responsive Design:** Ensure that the notifications are displayed responsively on different screen sizes and devices. Tests for the notification placement and appearance on desktop, tablet, and mobile views.
3. **Localization Support:** If the application supports multiple languages, ensure that the notification messages can be easily localized without affecting the layout.
4. **Code Implementation:** Integrate the notifications into the existing codebase using appropriate front-end technologies (e.g., HTML, CSS, JavaScript). Maintain clean and organized code structure while adding the notification functionality.