Documentation: Data Validation System Using Java Swing

1. System Design and Architecture:
The system is designed using Java Swing components to create a simple and user-friendly form where users can enter details such as their name, email, phone number, and password. The form is organized in a layout which arranges the labels and input fields neatly in rows.

For each input field:

- Name Field: Allows the user to enter their name.
- Email Field: Allows the user to enter their email address.
- Phone Number Field: Allows the user to enter a 10-digit phone number.
- Password Field: Allows the user to enter a password.

The form also includes a submit button that triggers the validation process when clicked. An error message is displayed after entering message if the user enters invalid data in any field. We used the JLabel, JTextField, and JPasswordField components for the input fields and JButton for the submit button.

2. Data Validation Mechanisms:
Each field in the form has specific validation rules to ensure correct data entry with the

- Name Field being the name must be at least 3 characters long with a mechanism that checks the length of the input.
- Email Field validation rule being the email must follow the format example@example.com with the mechanism regular expression regex is used to match the correct email format.
- Phone Number Field validation rule phone number must be exactly 10 digits long with only numbers including a mechanism expression is used to ensure the input contains only digits and is 10 characters long.
- Password Field validation rule password must be at least 8 characters long and include at least one uppercase letter, one lowercase letter, and one digit with the mechanism expression ensures that the password meets the required complexity regex pattern used.

3. Implementation Details:

The validation system was implemented using real-time feedback KeyListener to listen to changes in the text fields as the user types. When the user types in a field, the validation rules are applied instantly and error messages are displayed if any of the rules are violated. For example is when the user enters an invalid email, the system checks the input against the regex pattern immediately and displays a message "Invalid email

format" next to the field. This feedback helping users correct their mistakes before submitting the form. The submit button is connected to an ActionListener which performs the final check on all fields when the user attempts to submit the form. If any field contains invalid data, the submission is blocked and an appropriate error message is displayed. This prevents the system from accepting incomplete or incorrect information.

4. Testing and Debugging Procedures

Testing was done manually by entering different combinations of valid and invalid data into each field. The following scenarios were tested:

- Name Field entering less than 3 characters (e.g., "Sr") shows an error "Name must be at least 3 characters long and Entering a valid name like "Alice" shows no error.
- Email Field entering an invalid email like "test" shows an error "Invalid email format and entering a valid email like "test@example.com" shows no error.
- Phone Number Field entering less than 10 digits or letters (e.g., "12345") shows an error: "Phone number must be 10 digits and entering a valid 10-digit number like "1234567890" shows no error.

Password Field entering a password without an uppercase letter, lowercase letter, or number (e.g., "password") shows an error: "Password must be at least 8 characters, with at least one uppercase, one lowercase, and one digit and entering a valid password like "Password123" shows no error.

Debugging:

Debugging involved ensuring that the real-time feedback system worked correctly. One common issue during development was the mismatch between the regex patterns and the input. For example, an incorrectly written regex for email validation caused the system to accept invalid email formats. This issue was fixed by refining the regex patterns. Additionally, error messages were initially not aligned properly with the input fields, so the layout was adjusted to ensure that messages appeared directly next to the relevant field, making the interface more user-friendly.

Conclusion:

This Java Swing application ensures that users enter valid data into the form before submission. By implementing real-time feedback and thorough data validation for each input field, the system helps users avoid mistakes and enhances the overall user experience. The use of regular expressions for input validation ensures that the data entered follows the required formats, while the real-time feedback makes the form more interactive and responsive.