**SDLC**

**1. Planning & Analysis**

Business requirements from client:

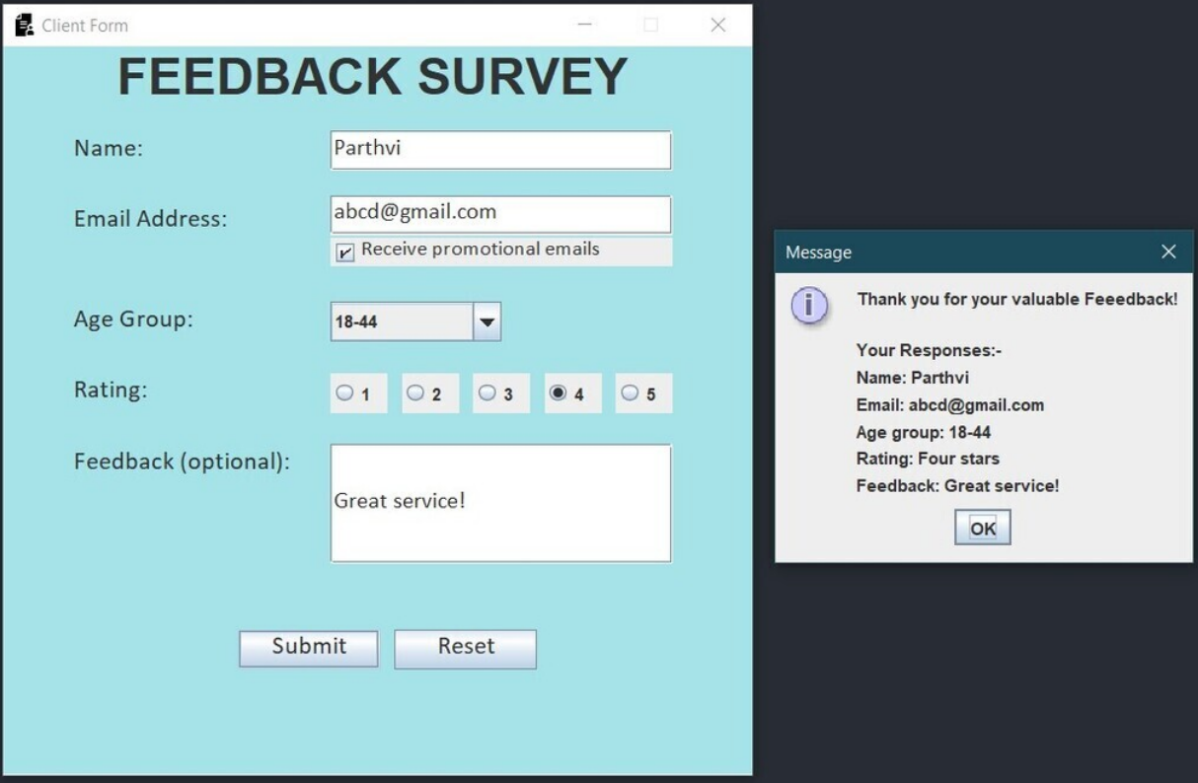
1. A GUI with labels, input text fields, a check box, a select dropdown, radio buttons, and a text area.
2. Client will connect to the server
3. Server will respond resending all the info to the Client
4. Client will display server’s response in a popup message that confirms all his/her inputs, separated by new lines and with an “OK” button to close the popup.

**2. Define Requirements**

**S**oftware **R**equirement **S**pecification (SRS):

1. User’s gui, as depicted in the sample Design image (from the web):
2. Labels for every text field input
3. A checkbox for “receiving promotional emails”
4. A dropdown list to select an age group
5. A set of radio buttons that will determine the “rating”
6. A feedback text area for customers to praise or complain
7. A submit button that will:
   1. Translate the inputs to JSON (for its readability and ease of use with various data types). It requires a Jar file (json-20230618.jar) added to the IntelliJ Library.
   2. Connect to the server via a socket, buffer reader and print writer.
   3. Read the response from the server into a Message box on the client side (as depicted).
8. A reset button to clear the form.

**3. Design**

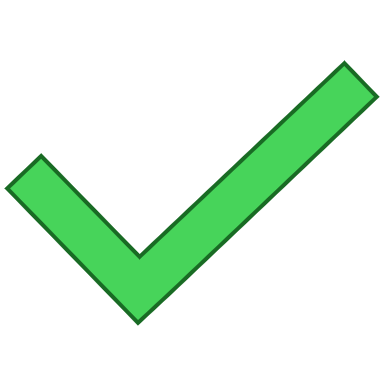
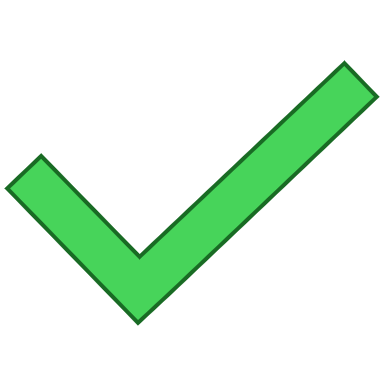
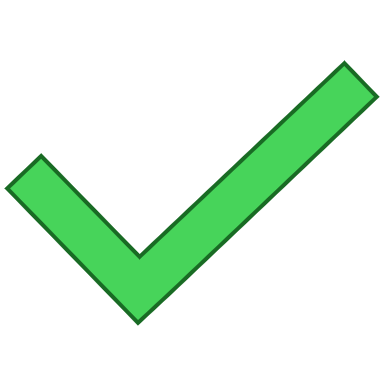


**4. Development**

Software modules:

1. ***GUI***: the client method will define labels, input text fields, a check box, a select dropdown, radio buttons, and a text area.
   1. Can send the strings as a JSON file through a PrintWriter defined as “output.”
   2. Send the JSON data to the server to receive a response. (Server Connection: Define this in a separate class (Values.java). Use “localhost” to connect locally).
2. ***Server***: will receive and resend the same info back to client for confirmation and display.
   1. The server will collect the JSON data, process it, and send a response back to the client. The client will then display this response in a popup message box.
3. ***Button***:
   1. A Submit button with an ActionListener for sending the info to the server and at the same time displaying a message box with the confirmation of the sent information.
   2. A Reset button to clear the form.
4. Display final Message box and **Close** connection.
   1. Read the server response and display it in a popup message box.
   2. Close buffer reader (input) and print writer (output) as well as the connection to the server. While this is closing it sends a “q” (quit) message to the server to close the server as well.

**5. Testing**

1. **Performance testing:**
2. **Functional testing:** Software meets the requirements
3. **Security testing:** Identifies potential vulnerabilities and weaknesses (possible crashes)
4. **Unit-testing:** Tests individual units or components of the software (Input validation)
5. **User acceptance testing:** Evaluates the software's user interface and overall user experience

**6. Deployment**

1. Release Candidate: Zip your client and server code and submit them on Canvas

5. General Availability: Attach a screenshot of your client GUI as well with a server answer in it.

6. Production Release

1. Regular Stable releases

**7. Maintenance**

1. Continuously monitor for vulnerabilities.
2. Update software with security patches.