Connecting a frontend developed with HTML, CSS, and JavaScript to a backend developed with Spring Boot can be accomplished through several methods. Here's a general outline of the steps and techniques you may consider:

.

API Creation:

.

Develop a RESTful API or a GraphQL API on your Spring Boot backend. This API will expose endpoints that your frontend can interact with to send or retrieve data.

Use Spring's annotations like `@RestController`, `@RequestMapping`, `@GetMapping`, `@PostMapping`, etc., to create your API endpoints.

.

CORS Configuration:

.

If your frontend and backend are hosted on different domains, you'll need to configure Cross-Origin Resource Sharing (CORS) in your Spring Boot application to allow requests from your frontend domain.

You can do this using Spring's `@CrossOrigin` annotation or by configuring a `WebMvcConfigurer`.

.

HTTP Client:

.

On the frontend, use an HTTP client like Fetch API, Axios, or jQuery AJAX to send HTTP requests to your Spring Boot backend.

Structure your frontend code to make requests to the backend, handle responses, and update the UI accordingly.

.

JSON Data Format:

.

Typically, data is sent between the frontend and backend in JSON format. Ensure your Spring Boot application is configured to handle JSON data, which is usually done by including the Jackson library (often included by default with Spring Boot).

.

WebSocket (Optional):

.

If your application requires real-time functionality, consider setting up a WebSocket connection between your frontend and backend.

Spring has support for WebSocket communication which can be set up to communicate with your frontend JavaScript code.

.

Error Handling:

.

Implement error handling on both the frontend and backend to manage situations where the API calls fail, or data is not returned as expected.

.

Authentication & Authorization:

.

If your application requires user authentication, implement an authentication and authorization strategy, such as JWT (JSON Web Tokens) or OAuth 2.0.

.

Testing:

.

Thoroughly test the communication between your frontend and backend to ensure data is being sent, received, and processed correctly.

.

Documentation:

.

Document your API endpoints, request/response formats, and any authentication requirements to provide clear instructions for frontend developers on how to interact with your backend.

.

Deployment:

.

When you're ready to deploy, ensure your deployment environment is properly configured to handle the communication between frontend and backend, including any necessary CORS configurations, firewall rules, and HTTPS setup.

These steps provide a solid foundation for connecting a frontend developed with HTML, CSS, and JavaScript to a backend developed with Spring Boot.