Must haves:

1. The system must have an easy-to-navigate drag-and-drop interface for connecting systems with Insocial intuitively by creating control flow graphs using nodes translated from company endpoints.
2. The user must be able to edit, disable, or enable created flows.
3. The system must support graphical representation of API connections.
4. The system must allow the creation of customisable API connections in the flow builder by enabling users to define nodes that can execute arbitrary API calls. These nodes should allow users to specify the endpoint, request method (e.g., GET, POST, PUT), and request body as needed for the desired API connection.
5. The system must integrate with the Insocial platform and its existing MySQL database.
6. The system must enable easy integration with existing authentication of Insocial platform after the development process is finished and use hardcoded bearer token during development process.
7. The system must be able to perform GET, POST, PUT actions on any given endpoint and trigger actions based on input received from API calls.
8. The system must provide support for logic flow and control, including triggers, conditions, schedules, and possibly loops.
9. The system must create user profiles that store and update graphs in the database.
10. The user must be able to execute the flow manually.
11. The backend must be implemented using PHP 8.
12. The database must use MySQL.
13. The frontend must be implemented using Angular CLI with TypeScript.
14. The system must execute the flow using the schedule specified by the user.
15. The system must support the JSON data format.
16. The product must be well documented, including source code, README files, and information about dependencies, build processes, APIs, and other relevant information.

Should haves:

1. The system should have an error handling system that allows users to see what and where in the flow a problem occurred.
2. The system should support "testing functionality" for users to check control flow outcomes and endpoint responses.
3. The system should match the design style used in Insocial, following the brand book for components and colours.
4. The system should have the ability to be translated into Dutch in later stages of development.
5. The system should be able to reuse nodes for multiple graphs created by the user.
6. The system should provide a pre-built template for a node specifically designed to handle Mail API calls.

Could haves:

1. The system could support debugging functionality.
2. The system could XML data formats.

3. The system could allow users to test API responses to check data format.