

About SendX:

We are a small, effective & profitable bootstrapped startup in the marketing tech space. We are a part of a much larger PE firm - [Inturact Capital](#).

We run multiple products in the email space - <https://sendx.io>, <https://sendpost.io> and <https://sendverify.io>.

We send 100's of millions of emails per month, process billions of events and write/read terabytes of data with a 6 member engineering team (of which 3 are from IIIT-A). SendX is the right place for folks who want to build things and work on interesting engineering problems with high ownership and a bias for action.

—

The assignments are real world projects which will resemble the type of problems you will face while working with us. You are free to use google, blogs, books, chatgpt or refer to other open source projects while attempting the problem. What is not okay is if you use the code as it is or are found copying the work done by the other candidates 😊 In such a case both of you will be disqualified since integrity is one of our core values.

About the assignment:

- You can attempt **either the frontend heavy or the backend heavy assignment but not both**.
- You can attempt and complete the assignment till **24th October midnight**
- The backend heavy assignment needs to be attempted in golang and the frontend heavy in vue javascript framework. We are trying to assess your ability to learn new things with this as a constraint.
- Every assignment has 3 sections - required, good to have and great to have. For you to be considered for the next stage you should be able to do the required section at the very least.
- Your code needs to be pushed to github in a public repository. The name for the project needs to be - **sendx-backend-<your_iiita_id> or sendx-frontend-<your_iiita_id>**. Your last commit id till 24th October midnight will be considered for the evaluation. If your code is not pushed to github in a public repository then it won't be considered for the evaluation
- How well your code is structured, good domain modeling, concise documentation and error handling will hold a lot of value
- Once the assignment is completed please share a 2-5 video demoing the project and explaining it at a high level. The video can be uploaded on youtube as a private video or to dropbox or google drive. The uploaded file link should be added to your **project readme**. This will be used to evaluate your communication skills and ability to work in a

team context.

Resources:

We have shared relevant resources which should be sufficient while attempting the assignments.

Frontend CSS / Design:

<http://tailwindcss.com/>

<https://tailwindui.com/>

Frontend Framework:

<https://vuejs.org/>

<http://vueschool.io/>

Backend Language:

Golang & WebApp

<https://www.golang-book.com/>

<https://astaxie.gitbooks.io/build-web-application-with-golang/>

Concurrency Patterns

<https://blog.golang.org/io2013-talk-concurrency>

<https://blog.golang.org/pipelines>

<https://blog.golang.org/waza-talk>

Backend Heavy:

You need to create a single HTML page with a search bar where users can enter the URL that they want to be crawled and click on the crawl button.

When the user does that the server checks if the URL has been crawled in the last 60 mins. If it crawled then the crawled page stored on the disk is read and returned back to the user.

In case the server does not have this page then it crawls in real time and returns back the page to the user.

Required:

These are the following things to consider:

- Pages at times may not be available so you may have to retry
- The application may have paying customers and non paying customers. So your server needs to always give priority for crawling to paying customers. For simplicity you can differentiate between paying and non paying customers via query parameter being passed to the backend from the frontend API call

Good to have:

- The crawler can crawl multiple pages concurrently. So you may have multiple workers that can crawl for maximum throughput. Ensure that the crawl URL can be picked by any worker. For simplicity you can assume that there are 5 crawler workers for paying customers and 2 for non paying customers.

Great to have:

If you are able to complete the above things then you should attempt the below items:

- The server also exposes two APIs which can be used to control
 - the number of crawler worker
 - the crawling speed per hour per worker

So the admin can say that our server should have 10 workers that can only crawl 100 pages per hour and if that limit is exceeded then the server returns an error saying that hourly crawl limit is exceeded.

The number of workers and number of pages crawled per hour can be changed at any point of time using the API endpoint.

Frontend Heavy:

You need to create a single page vue-app which has the similar navigation bar and set of screens as shown below. These screens are used to create an embedded form in one of our products.

Required:

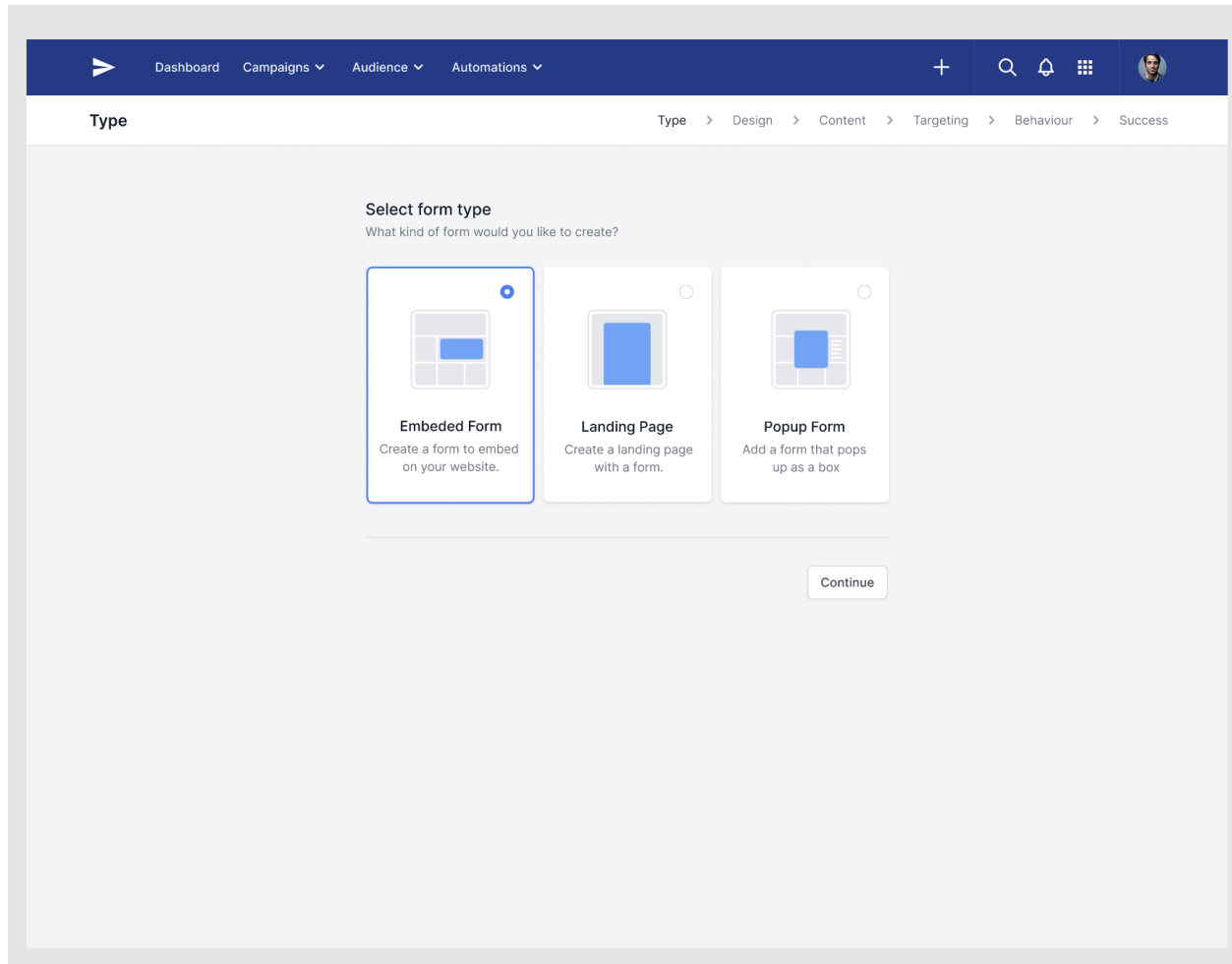
You need to create a largely pixel perfect design. You can use dummy images (placeholder images) for screen 2 & 3 (the design & content screen) for simplicity.

Good to have:

It will be better if you can create a simple API endpoint to which the entire configuration which the user selected can be sent via HTTP Post request.

Great to have:

Instead of using dummy images for screen 2 & 3 if you can integrate with <https://github.com/unlayer/vue-email-editor> and use actual HTML for at least 1 form design which opens up in the content screen



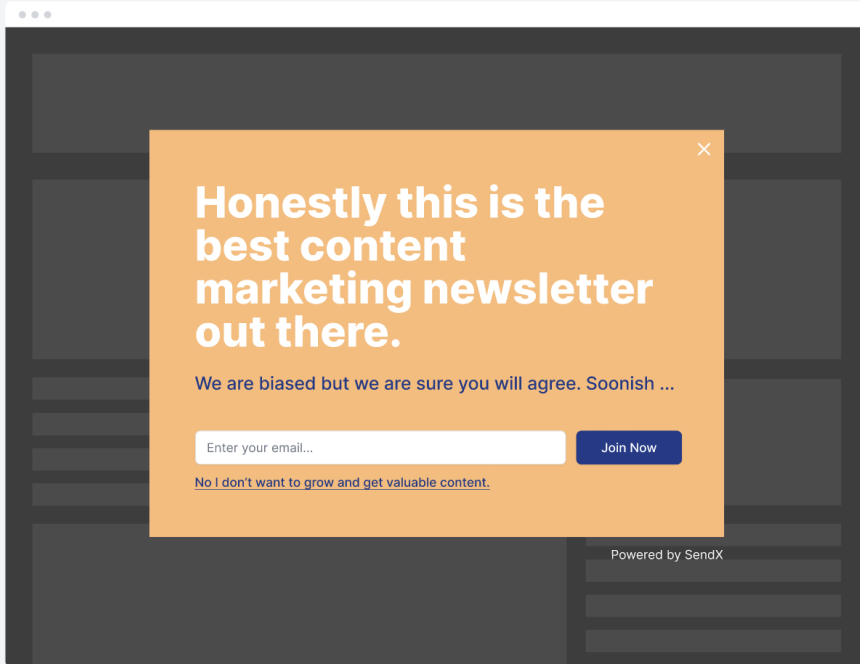
Success

Slider Modern

< Back

New (25 Feb 2022, 14:20) 

Save & Continue



Content



TITLE



TEXT



IMAGE



BUTTON



DIVIDER



SOCIAL



HTML



VIDEO



ICONS



MENU



STICKERS



GIFS

Position



Animation



Pop up

Success

Show the popup when visitor:

- ☒ Enters website
- ☐ Exits website
- ☐ On page scroll
- ☐ After seconds on website

[Advanced](#) ▾

[Go back](#)

Continue

Show the popup when visitor:

- ☒ Enters website
- ☐ Exits website
- ☐ On page scroll
- ☐ After seconds on website

[Advanced](#) ^

How often to show the popup

Set the period when you want to show the popup to the same visitor if they didn't subscribe.

- ☒ Every page view
- ☐ Every new browser session
- ☐ After days

When to stop showing the popup

- ☒ Never
- ☐ If the visitor has successfully completed the action
- ☐ If the visitor has successfully completed the action or the pop-up has been shown times.

[Go back](#)[Continue](#)

Dashboard

Campaigns

Audience


Automations

+

🔍

🔔

🗖



Success

Type > Design > Content > Targeting > Behaviour > Success

What happens after displaying success screen?

This is the screen visitors see after successfully submitting your form.

Do nothing

Close pop up after

5

seconds

Redirect to URL

Go back

Finish