Product Engineer - Design Challenge V3

Objective

Design the *Campaign Executor* that executes a given campaign for reaching out to candidates based on user-defined templates.

Background

Our system automatically identifies candidates who are a good fit for specific roles based on predefined criteria. Once a candidate is recognised as suitable, We initiates a personalised outreach campaign, termed an "outreach drip-campaign".

This campaign consists of a sequence of personalised messages sent over multiple days and through various channels, such as email and LinkedIn. The sequence of these messages is determined by user-defined templates, which are linked directly to specific roles.

When a candidate responds to any message in the campaign, the sequence is halted, reflecting a potential engagement. Our system manages these interactions, ensuring campaigns cease after 30 days if no response is received or continue monitoring for responses if all messages are sent before the 30-day mark.

The Campaign Executor capability, long requested by many users, aims to streamline recruitment efforts by automating candidate engagement. This enhancement is expected to increase response rates and efficiency while reducing the manual workload on recruiters.

Example User-defined Template:

1. Candidate identified -> LinkedIn request and email sent immediately.
2. Follow-up email sent after 2 days.
3. Final follow-up email sent after 5 days.

Task

<aside> 👆 Once a candidate is identified as a good fit, our system will create a personalised campaign definition and submit it to following endpoint exposed by **your system: *Campaign Executor.*** We don’t expect you to design other parts of the system that is not ***Campaign Executor***

</aside>

Create the Campaign Executor that manages and executes these outreach campaigns.

Endpoint for Campaign Executor:

- `PUT role/:roleId/candidate/:candidateId/campaign`
  - Accepts a campaign definition and starts the campaign immediately. There can only be 1 campaign running for a given candidate.
  - Returns a campaign ID.

Campaign Definition Example:

```
{
   "nodes": [
      {
         "id": 1,
         "type": "message",
         "channel": "LINKEDIN_MESSAGE",
         "message": "Hi Milan, \\nI'm interested in you background as Senior Software Engineer at Showpad. I'm looking for people like you for our role of Software Engineer with Startup Experience. \\nWould love to connect."
      },
      {
         "id": 2,
         "type": "message",
         "channel": "GMAIL",
         "subject_line": "Open to new opportunities?",
         "message": "Hi Milan,\\nI came across your profile and was impressed with your work as a Senior Software Engineer at Showpad. We're seeking individuals with your background for a Software Engineer role in our team.\\n Would you be open to discussing this opportunity?\\n\\nBest,Andreas"
      },
      {
         "id": 3,
         "type": "delay-step",
         "delaySeconds": 172800
      },
      {
         "id": 4,
         "type": "message",
         "channel": "GMAIL",
         "subject_line": "Re: Open to new opportunities?",
         "message": "Hi Milan,\\nI wanted to follow up on my previous email regarding the Software Engineer position at our company. We're eager to find a candidate with your startup experience and believe you could be a great fit. Could we arrange a short call to discuss this further?\\n\\n\\nBest,Andreas",
      },
      {
         "id": 5,
            "type": "delay-step",
            "delaySeconds": 432000
        },
        {
            "id": 6,
            "type": "message",
            "channel": "GMAIL",
            "subject_line": "Re: Open to new opportunities?",
            "message": "Hi Milan,\\nHope you're well. I haven't heard back from you regarding the Software Engineer position I messaged you about. We're finalizing our shortlist soon and would love to include you. Are you available for a quick chat this week to discuss the opportunity?\\n\\n\\nBest,Andreas",
        },
    ]
}
```

Deliverables:

1. **Wireframe:** Overview of candidates for a role, and being able to see the campaign state / progress. (Doesn't need to be pixel perfect, just a rough sketch.)
2. **Database Diagram**
3. **System Architecture:** Components, tools, and services used.
    1. Please list which frameworks, tools, services etc you'd like to use, if any.
4. **System Flow:** High-level process flow.

Considerations:

- 2, 3 and 4 are the most important, make sure you spend most time there.
- Speed and pragmatism are crucial; explain any scope reductions.
- Campaigns end automatically after 30 days or upon candidate response.
- Handle potential unavailability of communication channels.

Assumptions: You can assume everything that is not part of the system CampaignExecutor is available (so you don't have to focus on that part), for reference here are some endpoints that may be useful to model your solution:

(There is an authentication header that identifies the user in the system)

```
GET /user/me → User
```

```
 type User = {
        id: number
        firstName: string
        lastName: string
        email: string
}
```

-
- `GET /role/:roleId -> Role`

`GET /role -> Role[]`

```
 type Role = {
	id: number
	title: string
	description: string
}
```

-

`GET /role/:roleId/candidate → Candidate[]`

```
 type Candidate = {
	id: number
	firstName: string
	lastName: string
	email: string | null
	linkedin: string | null
	cv: Resume
}

type Resume = unknown // structured CV
```

-
- `GET /channel → Channel[]`
  - Returns the connected channels of the user

```
type Channel = {
	id: number
	type: "EMAIL" | "LINKEDIN"
	status: "OK" | "ERROR" | "ERROR_CREDENTIALS" // (the user got logged out of the channel)
}
```

-

`POST /channel/:channelId/chat/:candidateId/message`

```
 type PostMessageBody = {
	body: string
	subject: string | null
}
```

```
type PostMessage200Response = {
        messageId: number
}
```

-

```
GET /channel/:channelId/chat/:candidateId/message → Message[]
```

```
 type Message = {
        id: number
        from: number // candidateId or userId
        to: number // candidateId or userId
        subject: string | null
        body: string
        sentAt: Date | null
        openedAt: Date | null
}
```

-