1. **Email campaign**

Email campaign is emailing schedule for sending the marketing email to customer. The campaign should send to multiple customer email and can be set the schedule time.

1. **Email campaign API design**
   1. Create new email campaign
      1. Information

|  |  |
| --- | --- |
| **Task** | Create new email campaign |
| **Business case** | When customer want to create new email campaign |
| **API flow** | 1. Customer submit create new campaign form 2. The email campaign created |
| **Sequence flow** | 1. Validate input 2. Insert data to database 3. Add email campaign schedule to the queue 4. Return http response with data object |

* + 1. Request/Response

|  |  |  |  |
| --- | --- | --- | --- |
| **API Name** | **Endpoint URL** | **Verb** | **Description** |
| Create email campaign API | http://api.customer/v1/email-campaign | POST | Create new email campaign |
| **Headers** | Content-Type: application/json Authorization: Bearer <jwt token> | | |
| **Request Body** | Request sample:  {  "campaign\_name": "Summer Promo",  "subject": "Hot Deals This Summer!",  "email": "customer01@email.com,customer02@email.com",  "content": "<div><div><h1>Welcome to Summer</h1></div><div><img src=\"https://img.customer.com/campaign/hotdeal.jpg\" alt=\"Hot deals this summer\" /></div></div>",  "status": "active",  "schedule": "2025-04-19T08:00:00.000+07:00"  } | | |
| **Response Body** | Response sample – 201 Created:  {  "id" : 1,  "uuid": "322f0b2e-9335-4414-aaf8-e28786b0f2b0",  "campaign\_name": "Summer Promo",  "subject": "Hot Deals This Summer!",  "email": "customer01@email.com,customer02@email.com",  "content": "<div><div><h1>Welcome to Summer</h1></div><div><img src=\"https://img.customer.com/campaign/hotdeal.jpg\" alt=\"Hot deals this summer\" /></div></div>",  "status": "active",  "schedule": "2025-04-19T02:00:00.000Z",  "created\_at": "2025-04-18T18:41:19.129Z",  "updated\_at": "2025-04-18T18:41:19.129Z",  "version": 1  }  Response sample – 400 Bad Request:  {  "message": [  "subject must be a string",  "subject should not be empty"  ],  "error": "Bad Request",  "status\_code": 400  } | | |

* + 1. Input parameters

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Parameter** | **Data type** | **Mandatory** | **Description** | **Example** |
| 1 | campaign\_name | string | Y | Campaign name | Summer promotion |
| 2 | subject | string | Y | Email subject | Hot deal this summer |
| 3 | email | string | Y | Customer email, can be multiple that separate by , (comma) | customer01@email.com, customer02@email.com |
| 4 | content | string | Y | Email content that supports html | <div><h1>Welcome to Summer</h1></div> |
| 5 | status | string - active - inactive | X | Status for email campaign support only active or inactive  when empty system will set active | active |
| 6 | schedule | string | X | Schedule time to start message sending  when empty immediately sending after creating successfully | 2025-04-19T08:00:00+07:00 |

* + 1. Output parameters

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Parameter** | **Data type** | **Mandatory** | **Description** | **Example** |
| 1 | id | int | Y | ID number of email campaign | 1 |
| 2 | uuid | string | Y | UUID of email campaign | 322f0b2e-9335-4414-aaf8-e28786b0f2b0 |
| 3 | campaign\_name | string | Y | Campaign name | Summer promotion |
| 4 | subject | string | Y | Email subject | Hot deal this summer |
| 5 | email | string | Y | Customer email | customer01@email.com, customer02@email.com |
| 6 | content | string | Y | Email content that supports html | <div><h1>Welcome to Summer</h1></div> |
| 7 | status | string | Y | Status for email campaign | active |
| 8 | schedule | string | Y | Schedule time to start message sending | 2025-04-19T02:00:00.000Z |
| 9 | created\_at | string | Y | Time when created email campaign | 2025-04-18T18:41:19.129Z |
| 10 | updated\_at | string | Y | Time when email campaign updated | 2025-04-18T18:41:19.129Z |
| 11 | version | int | Y | Number of data changed | 1 |

* + 1. HTTP Reponse codes

|  |  |
| --- | --- |
| Response code | Description |
| 201 Created | Create new campaign success and return data |
| 400 Bad Request | Error occurs, read response body for details |

1. **Data validate and handling**

When API is called, first should validate input parameters like this.

* Campaign Name, Subject, Email and Content should not be empty because the data can’t be generated by the system
* Email should validate by regular expression that math the email pattern and the email can be support multiple email
* Content can be plain text or HTML. If the content is HTML, should test the content valid html format
* Status can be active or inactive only. If empty system generates active automatically
* Schedule can be empty, when empty system generates current time, when not empty check the format that math ISO8601

When validation failed, API would return 400 Bad request and response body as the cause of errors.

1. **How to enqueue sending email schedule**

As the requirement, customer need to send campaign via email to multiple customers. So, we should create the sub task for email sending by chunk.

Subtask should split customer email with proper chunk size. The chunk size can calculate by message sending provider. Ref to MailGun, support for sending 1,000 recipients at once.

1. **Messaging sending background process**

The process of email sending should create as separately background process. The process will execute every 1 minute, 5 minutes or needed by system engineer configure. So, the service should support for configure it.

When the process. Update campaign status as sending. Then sending message by queue. Email sending can run parallel subtask for make the process faster.

Sending result data should be collected as atomic of piece as we can, we should collect data sent to whom and failed or succeed status, for easily generate the report. And atomic result would make the system can track atomic progress. All of sub tasks done, update campaign status as done.