Terminal

anomify

Studying documentation perception of developers and its effect on learning, trust and brand value

Author: Shahab Khan
Supervisor: Dr Yoram Chisik

/ABSTRACT

This study was undertaken in collaboration with Anomify to investigate the obstacles encountered by developers in connecting their data to Anomify's service. Anomify is a SaaS product which uses ML techniques to automatically detect and alert anomalies in time-series data. This study explores the impact of software documentation, in particular, Anomify's API documentation on developer's understanding, experience, trust, and value perception. It also explores the existence of issues that could prevent developer's from successfully integrating their data with Anomify's service.

/DISCOVER

<Research Problem>

Data can be sent to Anomify in the following ways

- Time Series Database Connection
- HTTP API

HTTP API requires that the data is sent according to Anomify's requirements. Issues can arise when the data is not sent at regular intervals or of the correct format or requirement which are mentioned in Anomify's API documentation. It is critical that these requirements are conveyed in an effective and simplified manner to enable easy implementation by the developers.

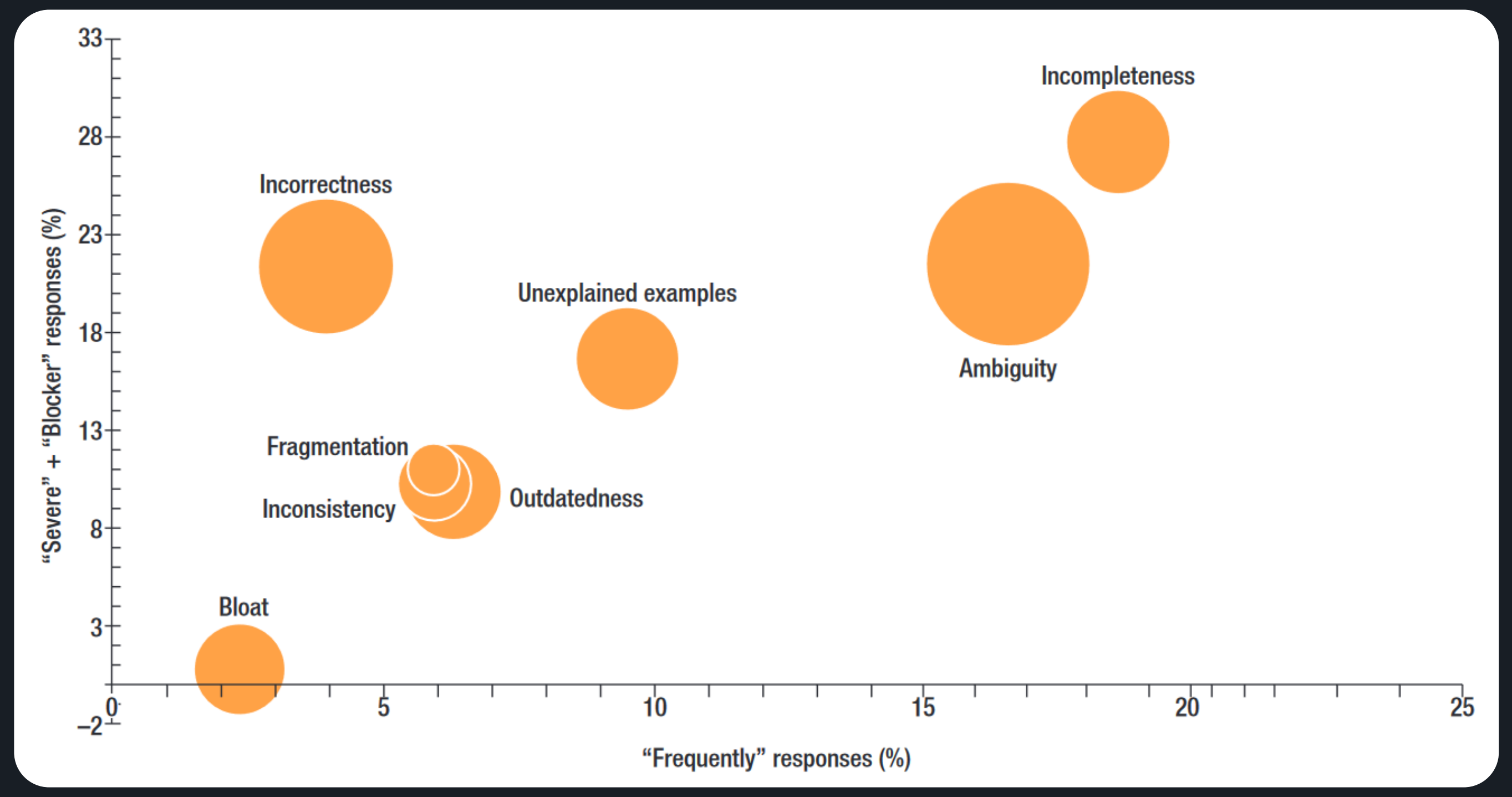
This is where the difficulty occurs, and this is where developers may be getting frustrated as there can be a very large variety of tech stack configurations.

<Literature Review>

Developers can play the following Key roles in adoption of a product : Initiator, Decision Maker, Business decision Maker, Influencer or Approver. Therefore, it is critical that the developer experience is positive to ensure adoption of the service/ product

Documentation can play a vital role in the adoption of a Software product. The most severe obstacles faced by the developers while learning and implementing new APIs consisted of learning content such as documentation as shown in the chart on the right

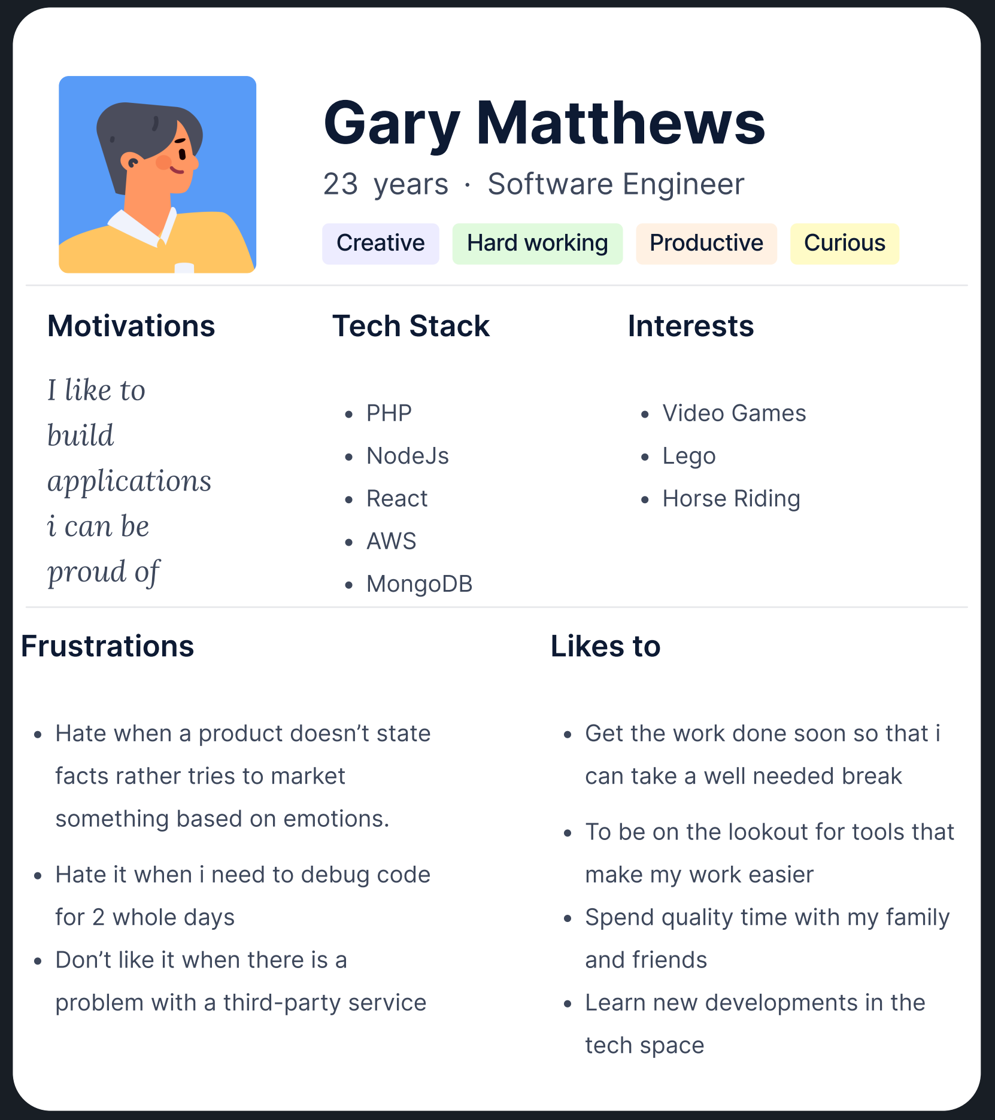
Problems with API Documentation



<Hypothesis>

The Documentation negatively affects the developer understanding , trust and value perception of the Brand.

Developer Persona



Gary Matthews

23 years · Software Engineer

Creative Hard working Productive Curious

Motivations

I like to build applications i can be proud of

Tech Stack

- PHP
- NodeJs
- React
- AWS
- MongoDB

Interests

- Video Games
- Lego
- Horse Riding

Frustrations

- Hate when a product doesn't state facts rather tries to market something based on emotions.
- Hate it when i need to debug code for 2 whole days
- Don't like it when there is a problem with a third-party service

Likes to

- Get the work done soon so that i can take a well needed break
- To be on the lookout for tools that make my work easier
- Spend quality time with my family and friends
- Learn new developments in the tech space

<Competitive Analysis>

Twilio: Twilio Docs allow developers to split the page into two parts, the left side is for documentation specifications and the right side contains code snippets which can be copied and edited regarding the API call in view. Twilio also contains Use-Case tutorials which show how Twilio was used by other developers.

Stripe: Stripe's API documentation is displayed in a sleek and minimal manner with no useless information. It contains code snippets on the right of every API call so that developers can copy the code directly.

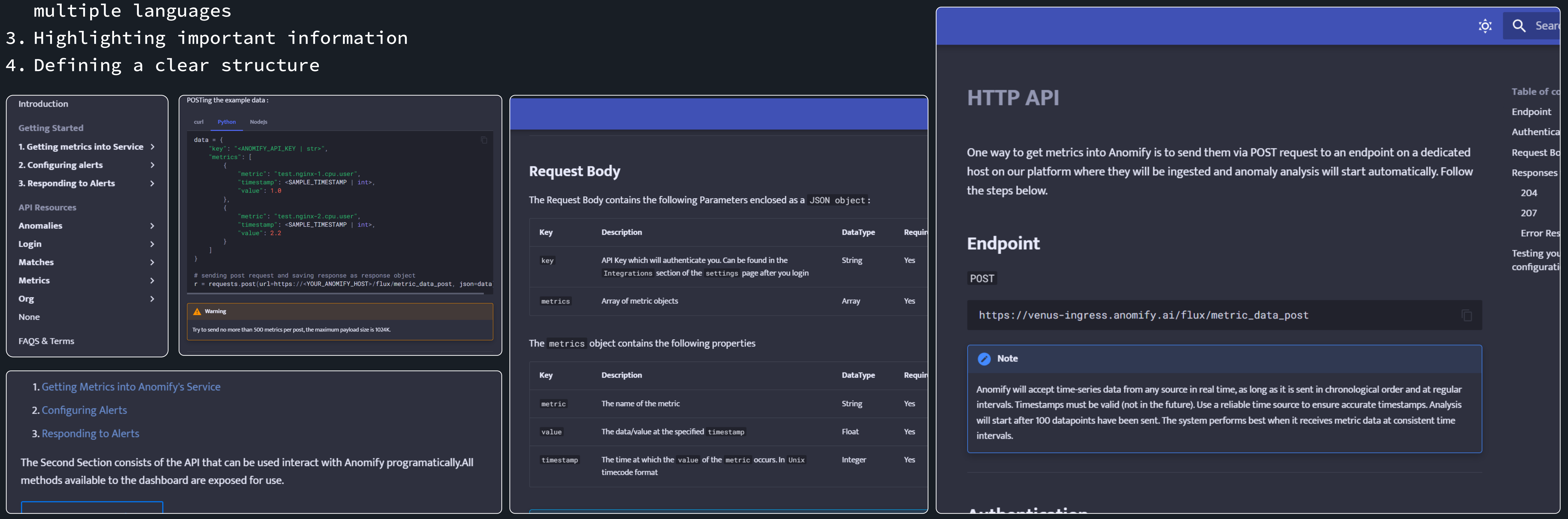
Dropbox: It is known for its simplicity; it first asks developers about their software/programming language and only shows documentation related to their programming language

It also allows developers to Interact with the API in real-time directly on the webpage without having to send any data or tokens from their software or programming language

Discord: Discord's documentation is very minimal and has in-depth and exhaustive documentation. It allows developers to quickly search for documentation with keyboard shortcuts. It also has a plethora of video content to ease understanding and learning for developers.

/DEVELOP

1. Combining both the documentation pages
2. Adding useful features such as hyperlinks and color coded code examples in multiple languages
3. Highlighting important information
4. Defining a clear structure



Introduction

Getting Started

1. Getting metrics into Service >
2. Configuring alerts >
3. Responding to Alerts >

API Resources

Anomalies >
Login >
Matches >
Metrics >
Org >
None

FAQS & Terms

POSTing the example data :

curl Python NodeJs

```
data = {
    "key": "<ANOMIFY_API_KEY | str>",
    "metrics": [
        {
            "metric": "test.nginx-1.cpu.user",
            "timestamp": <SAMPLE_TIMESTAMP | int>,
            "value": 1.0
        },
        {
            "metric": "test.nginx-2.cpu.user",
            "timestamp": <SAMPLE_TIMESTAMP | int>,
            "value": 2.2
        }
    ]
}

# sending post request and saving response as response object
r = requests.post(url=https://<YOUR_ANOMIFY_HOST>/flux/metric_data_post, json=data
```

⚠ Warning

Try to send no more than 500 metrics per post, the maximum payload size is 1024K.

1. Getting Metrics into Anomify's Service
2. Configuring Alerts
3. Responding to Alerts

The Second Section consists of the API that can be used interact with Anomify programatically.All methods available to the dashboard are exposed for use.

Request Body

The Request Body contains the following Parameters enclosed as a `JSON object`:

Key	Description	DataType	Requir
`key`	API Key which will authenticate you. Can be found in the `Integrations` section of the `settings` page after you login	String	Yes
`metrics`	Array of metric objects	Array	Yes

The `metrics` object contains the following properties

Key	Description	DataType	Requir
`metric`	The name of the metric	String	Yes
`value`	The data/value at the specified `timestamp`	Float	Yes
`timestamp`	The time at which the `value` of the `metric` occurs. In `Unix` timecode format	Integer	Yes

Sear

HTTP API

One way to get metrics into Anomify is to send them via POST request to an endpoint on a dedicated host on our platform where they will be ingested and anomaly analysis will start automatically. Follow the steps below.

Endpoint

POST

```
https://venus-ingress.anomify.ai/flux/metric_data_post
```

Note

Anomify will accept time-series data from any source in real time, as long as it is sent in chronological order and at regular intervals. Timestamps must be valid (not in the future). Use a reliable time source to ensure accurate timestamps. Analysis will start after 100 datapoints have been sent. The system performs best when it receives metric data at consistent time intervals.

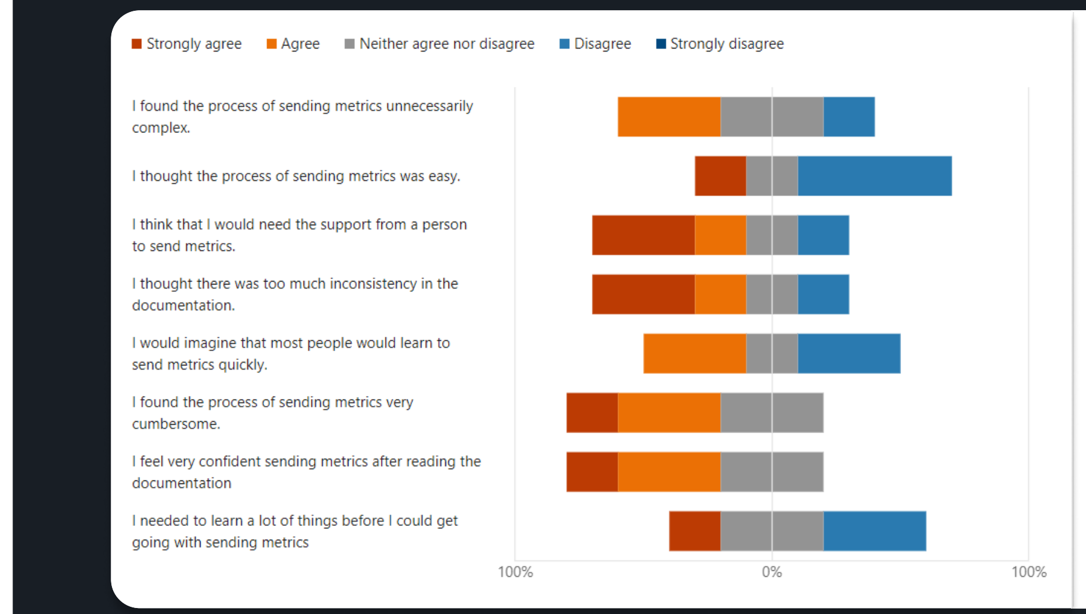
Authentication

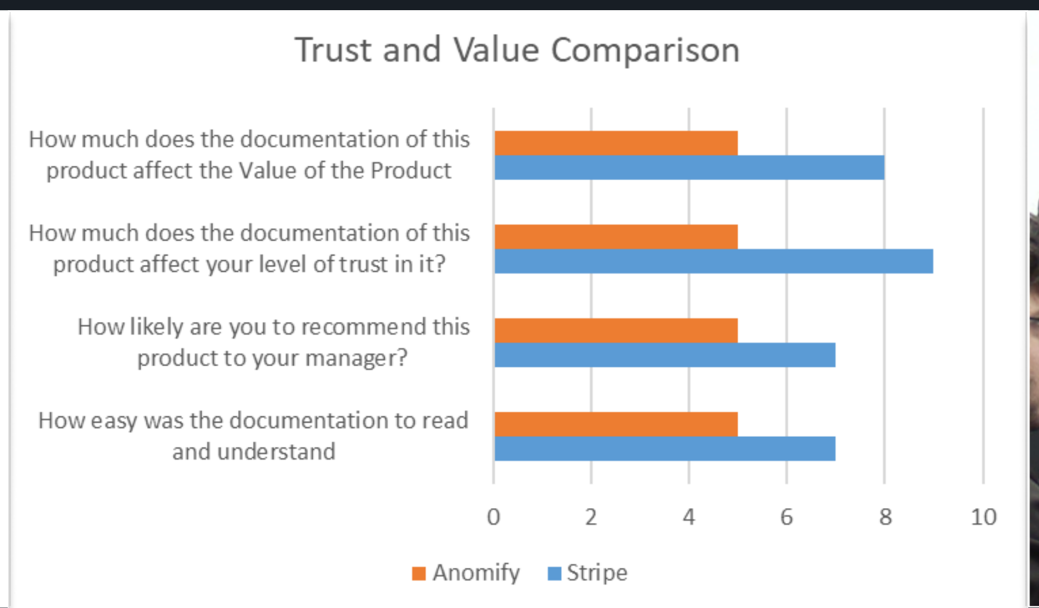
Table of co
Endpoint
Authentica
Request Bo
Responses
204
207
Error Res
Testing you configurati

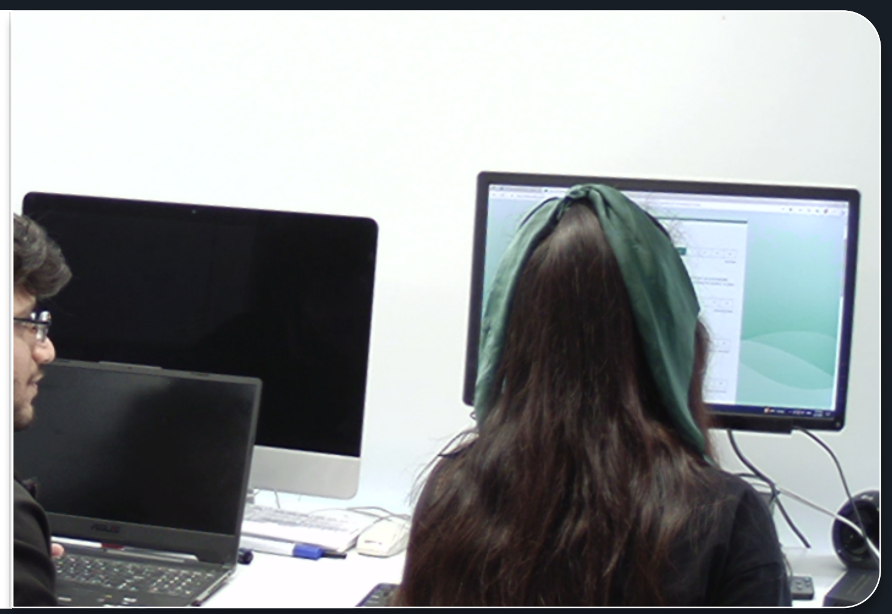
/DEFINE

<Primary Research>

Usability study, interviews and questionnaires were conducted to test the hypothesis and define the core issues that were being faced by the developers. These tests were also conducted to test the validity of the prototypes







Different Documentation Pages

Anomify's resources consisted of two separate documentation pages API Docs and Support Docs. All 5 Participants could not find the correct documentation page and needed help. They expected the resource to exist in API docs instead of Support Docs.

Incorrect and Incomplete

Some of the API responses in the API Documentation page had incorrect and incomplete responses. these responses showed disapproval and loss of trust. Evidence of this can be seen in the low scores given for trust and value in the product

Too much information

Some participants preferred to scan through the entire page for the required information rather than reading a body of text. This study found that too much bloat would cause participants to miss vital API information.

Code Examples in different programming languages

The feature of selecting their preferred programming languages for the code examples from Stripe documentation. It allowed them to quickly understand the code examples. This feature is missing in both of Anomify's documentations.

Navigation Index

The Taxonomy of the items in the navigation index confused participants. The participants could not figure out what the API endpoints in the navigation index meant. The navigation index did not work as intended as it lost the position and the page in which the participant was.

Low impressions of trust and brand value

The documentation negatively affected their perception of trust and brand value as seen from qualitative and quantitative results.

/DELIVER

<Evaluation>

The prototype was tested with the same usability study and questionnaire from the discovery phase. Interview questions were changed to explore their experience of the prototype rather than to explore their past experience with reading software documentation. The tests were carried out with new participants.

<Conclusion>

This study has found and solved some issues in the documentation which prevented developers from learning how to send data to Anomify's service. By doing this it has also improved feelings of trust and brand value of the product for the developer.

It has outlined a basic hierarchy for API documentation and also stressed the importance of highlighting vital API characteristics in the documentation and providing features such as code examples in multiple programming languages, both of which vastly improves the experience for developers. It has also provided indications of how developer experience can negatively affect adoption of the product.

<Future Work and Limitation>

This study tests the understanding of the API documentation rather than the implementation of the product. Although Eye tracking was planned for this study, it could not be achieved due to the online nature of the interviews. Developer Trust showed a small increase between Anomify's original documentation and the prototype. As Uncovered from the interviews, documentation could have a lesser effect on trust when compared to the company reputation.

Information Heirarchy

Participants expected the correct page to be under the API Resources section rather than Getting Started Section.

Participants got confused with the taxonomy

Some participants got confused between the taxonomy of Anomify's service such as metrics and anomalies

Easier to Understand

A majority of the participants agreed that they found the process of metrics easy after reading the prototype and that they were confident in their ability to send metrics

Useful Features

Highlighting important API information via a clear hierarchy, tables, color coded code examples for various programming languages and colorful implementation details enabled users to scan important information quickly

Better impressions of trust and brand value

There is a significant increase in the ease of understanding, perception of value and likeliness of recommendation in the prototype.