

Master's Programme in Computer, Communication and Information Sciences

Development of A/B testing capability for digital service operations using a third-party tool

Siiri Sääksvuori

© 2024

This work is licensed under a [Creative Commons](https://creativecommons.org/licenses/by-nc-sa/4.0/) “Attribution-NonCommercial-ShareAlike 4.0 International” license.



Author Siiri Sääksvuori

Title Development of A/B testing capability for digital service operations using a third-party tool

Degree programme Computer, Communication and Information Sciences

Major Computer Science

Supervisor Prof. Marko Nieminen

Advisor Janne Villikka (M.Sc)

Collaborative partner Helen Oy

Date 13 June 2024

Number of pages 70+8

Language English

Abstract

High competition exists between digital services and in agile development new features and versions of the services are constantly released. The services must stand out in terms of offered functionalities and user experience. However, as much as 90% of the time the intuition of development organisations on user preferences is incorrect.

To combat this, the industry giants run online controlled experiments, including A/B testing, with real users. In A/B testing the users of the service are divided into two groups, where one group is shown the original version of the service, and the other a modified one. Statistical methods are used to study whether there exists any user behaviour difference between the two groups.

Running these experiments requires a specific setup which development in-house can demand a large amount of resources. In recent years, the number of available A/B testing tools offered by third-party companies has increased. These tools decrease the experimentation costs and make the testing easier which expands the experimentation possibility to smaller organisations.

In this thesis A/B testing capability is developed for a development process using a third-party tool. The development process is used to develop a digital service for the customers of a Finnish energy company. The capability is developed for the web front-end of the service. However, possible desire to expand the capability to back-end and mobile is considered throughout the thesis.

The A/B testing tool is selected using a systematic comparison derived from literature. The initial comparison includes six tools from which AB Tasty, Optimizely and Split support the required technologies, and features. More in-depth comparison is done between them. Split is chosen as the most suitable tool for the development process as it offers the most extensive features and is already partly used in the project.

The tool is validated using A/A tests and an A/B testing experiment is run to develop and test the A/B testing capability. In the experiment the position of an element in the landing page is moved. The experiment runs for seven days in production. Thus, the results are based on real user behaviour. The experiment is successful, and it confirms that A/B testing capability is developed for the development process.

Keywords A/B testing, split testing, online controlled experiments, A/A testing

Tekijä Siiri Sääksvuori

Työn nimi A/B testaus kyvykkyyden kehittäminen digitaaliselle palvelutoiminnalle käyttäen kolmannen osapuolen työkalua

Koulutusohjelma Computer, Communication and Information Sciences

Pääaine Computer Science

Työn valvoja Prof. Marko Nieminen

Työn ohjaaja FM Janne Villikka

Yhteistyötaho Helen Oy

Päivämäärä 13.6.2024

Sivumäärä 70+8

Kieli englanti

Tiivistelmä

Kilpailu digitaalisten palveluiden välillä on kovaa ja ketterän kehittämisen myötä uusia ominaisuuksia ja versioita palveluista julkaistaan jatkuvasti. Palveluiden tulee erottua edukseen sekä tarjottujen toiminnallisuuksien että käyttäjäkokemuksen perusteella. Kuitenkin jopa 90% ajasta ohjelmointikehitysorganisaatioiden intuitio käyttäjien mieltymyksistä on väärä.

Alan jättiläiset ovat ratkaisseet tämän ajamalla verkossa ohjattuja kokeiluja, joihin myös A/B-testaus kuuluu. A/B-testauksessa palvelun käyttäjät jaetaan kahteen osaan, joista toiselle esitetään alkuperäinen versio palvelusta ja toiselle uusi versio. Testissä tarkastellaan statistisilla menetelmillä vaikuttaako muutos käyttäjien käyttäytymiseen.

Näiden kokeilujen ajaminen vaatii tarkasti kehitetyn ratkaisun, jonka kehittäminen itse vaatii valtavia resursseja. Viime vuosina kolmansien osapuolten tarjoamien A/B-testaustyökalujen määrä on kasvanut merkittävästi. Nämä työkalut tarjoavat halvemman ja helpomman tavan testata, mikä mahdollistaa A/B-testauksen myös pienemmissä yrityksissä.

Tässä työssä kehitetään A/B-testauskyvykkyys käytössä olevalle kehitysprosessille kolmannen osapuolen työkalun avulla. Tällä kehitysprosessilla kehitetään digitaalista palvelua energiayhtiön asiakkaille. Työ on rajattu kehittämään A/B-testauskyvykkyys palvelun verkkosivun front-endille. Työssä otetaan kuitenkin huomioon mahdollinen tarve laajentaa testaus myös back-endin ja mobiilin puolelle.

A/B-testaustyökalu valitaan systemaattisen vertailun avulla. Alustavassa vertailussa on mukana kuusi työkalua. Näistä AB Tasty, Optimizely ja Split tukevat tarvittavia teknologioita ja ominaisuuksia, joten ne valitaan tarkempaan vertailuun. Työkaluista Split on sopivin työkalu kehitysprosessille, sillä se tarjoaa laajimmat ominaisuudet ja on jo osittain käytössä kehitysprosessissa.

Splitin toimivuus varmistetaan A/A-testin avulla. Työkalulla toteutetaan A/B-testauskokeilu, jossa etusivun elementtejä siirretään. Kokeilua ajetaan seitsemän päivän ajan tuotannossa, jolloin kokeilun data perustuu oikeisiin käyttäjiin. Kokeilu on onnistunut ja sillä voidaan todentaa että A/B-testauskyvykkyys on kehitetty kehitysprosessille.

Avainsanat A/B-testaus, split testaus, kontrolloidut verkkokokeilut, A/A-testaus

Acknowledgements

I would like to thank Helen Oy for making the writing of this thesis possible. And thank you to the people from Helen, especially Janne Villikka, who participated in and followed this thesis process. Furthermore, special thanks to Marko Nieminen for supervising this thesis and giving great suggestions along the way.

In addition, I want to thank the friends I made during my studies in Aalto for the support during the years of studying while making that time fun and educational. Furthermore, thank you to my family for supporting me during my whole educational journey. And finally, thanks to my amazing partner for supporting and cheering me on throughout this thesis process and my studies and especially thank you for helping me proofread this paper.

Contents

Abstract	3
Abstract (in Finnish)	4
Acknowledgements	5
Contents	6
Abbreviations	9
1 Introduction	10
1.1 Thesis objective and research questions	11
1.2 Structure of the thesis	12
2 A/B testing	13
2.1 Use cases	13
2.2 Benefits and limitations	14
2.3 Metrics	15
2.4 Statistics	16
2.5 Testing tools	17
2.6 Variants	19
2.7 Implementation	19
2.8 Validation	20
2.9 Ethical questions	21
3 Development process and product description	23
3.1 Development process	23
3.2 Product	23
3.2.1 Users	24
3.2.2 Goals	24
3.2.3 Technologies and practices	24
3.2.4 Restrictions	25
4 Methodology	26
4.1 Tool comparison and selection	26
4.2 A/B testing experiment	27
5 Identifying Assessment criteria and tools	29
5.1 Selecting an initial list of tools	29
5.2 Identifying functionalities and restrictions	29
5.3 Initial comparison and elimination	30
5.4 Identifying assessment criteria	31

6	Tool comparison and selection	33
6.1	AB Tasty	33
6.1.1	Help documentation	33
6.1.2	Events & Metrics	33
6.1.3	Running an A/B test	34
6.1.4	Data visualisation & analytics	34
6.1.5	Effects on performance	35
6.1.6	GDPR compliance & consent	35
6.2	Optimizely	35
6.2.1	Help documentation	35
6.2.2	Events & Metrics	35
6.2.3	Running an A/B test	36
6.2.4	Data visualisation & analytics	36
6.2.5	Effects on performance	37
6.2.6	GDPR compliance & consent	37
6.3	Split	37
6.3.1	Help documentation	37
6.3.2	Events & Metrics	37
6.3.3	Running an A/B test	38
6.3.4	Data visualisation & analytics	38
6.3.5	Effects on performance	39
6.3.6	GDPR compliance & consent	39
6.4	Comparison	39
6.4.1	Help documentation	39
6.4.2	Events & Metrics	39
6.4.3	Running an A/B test	40
6.4.4	Data visualisation & analytics	41
6.4.5	Effects on performance	41
6.4.6	GDPR compliance & consent	41
6.5	Results	42
7	A/B testing experiment	43
7.1	Preliminary actions	43
7.1.1	Legal approval	43
7.1.2	Modifying cookies	43
7.1.3	Identifying metrics and testable area	43
7.2	Tool deployment	45
7.2.1	Important concepts	45
7.2.2	Choosing the SDK	45
7.2.3	Installing and initialising the SDK	46
7.2.4	Creating and using feature flags	47
7.2.5	Sending event data	47
7.3	Metric creation	49
7.4	Validation	49
7.5	Constructing a hypothesis	52

7.6	Creating the treatment variant	52
7.7	Running the A/B test	52
7.8	Analysing results	53
7.8.1	Observations	54
8	Discussion	56
8.1	Research questions	56
8.1.1	What is the development process in use and how A/B testing with a third-party tool can be incorporated with that?	56
8.1.2	What aspects should be considered when choosing a tool for the development process?	56
8.1.3	How tools for A/B testing differ from each other and what tool is the most suitable for the process?	57
8.1.4	What steps are needed for implementing A/B testing in practice?	58
8.1.5	Is the chosen tool suitable to use for the A/B testing in the development process?	58
8.2	Development of A/B testing capability	59
8.3	Future of A/B testing and online experimentation	59
9	Conclusion	61
	References	63
A	A/A test results	71
B	A/B test results	73
C	Quick guide to A/B testing	75
D	Running an A/B test with Split	76
E	What is A/B testing?	77

Abbreviations

AHP	Analytic Hierarchy Process
API	Application Programming Interface
CD	Continuous Deployment
CDN	Content Delivery Network
CI	Continuous Integration
CSV	Comma-Separated Values
DevOps	Development Operations
FAQ	Frequently Asked Questions
GDPR	General Data Protection Regulation
JSON	JavaScript Object Notation
KPI	Key Performance Indicator
NPM	Node Package Manager
OCE	Online Controlled Experiment
OEC	Overall Evaluation Criteria
PDF	Portable Document Format
PO	Product Owner
REST	Representational State Transfer
RUM	Real User Monitoring
SDK	Software Development Kit
UI	User Interface
UX	User Experience
VWO	Visual Website Optimizer
WYSIWYG	What You See Is What You Get

1 Introduction

High competition exists within the Finnish electricity sales market. The competition has expanded to companies offering more extensive, versatile, and user-friendly digital services for the customers. For instance, the three largest electricity sales companies in Finland are Fortum, Helen and Vattenfall (Jurvelin, 2024). All of them offer their own digital service; Fortum has Oma Fortum (Fortum, n.d.), Helen has Oma Helen (Helen, n.d.) and Vattenfall has Oma Energia (Vattenfall, n.d.).

The services are used as a marketing point. Thus, it is important for a company to ensure that the changes made to the digital service indeed improve the service and help to achieve organisational goals. This applies to any companies developing online services. However, when developing new features and changes to the service, as much as ninety percent of the time the company's intuition on user preferences are incorrect (Fabijan, Dmitriev, Olsson, & Bosch, 2017).

To test and improve their digital services, software industry giants have started running online controlled experiments (OCEs) (Larsen et al., 2023). The OCEs allow evaluating the service with its actual users and can be considered as a digital version of randomised controlled trials (Larsen et al., 2023). A/B testing is one type of an OEC (Kohavi & Longbotham, 2023).

In A/B testing process the performance difference between a control variable (A) to an experimental variable (B) is statistically compared by displaying the variables randomly to the end users (Kompella, 2015) (Martin, 2015). Running an A/B testing experiment requires a specific solution and building it in-house involves a large amount of engineering effort, therefore it is not a viable option for companies with smaller resources (Siroker, Koomen, & Harshman, 2013). However, in recent years the amount of available A/B testing tools has increased; thus, making the testing process easier, faster, and more cost effective for different sized companies (Koning, Hasan, & Chatterji, 2022).

The benefits of A/B testing can be substantial for organisations of varying sizes. For instance, start-ups can double their performance and industry giants can gain hundreds of millions of additional revenues by adopting A/B testing as part of the development process (Koning et al., 2022). In addition to maximising their revenues, organisations can use A/B testing to optimise user experience (UX) (Larsen et al., 2023). Good UX is important as the competition between the online services is not limited to functionality of the service but how the contents are presented (Martin, 2015).

Thus, motivated by the possible revenue increase and advantage on creating better digital service for the customers, A/B testing capability is developed for an electricity sales company in this thesis. More specifically, the capability is developed for a development process used in the organisation to develop a digital service for customers. The capability is developed by using a third-party A/B testing tool and this paper describes the process of choosing a suitable tool for the development process. Furthermore, an A/B testing experiment is conducted on the digital web service. The experiment is used to ensure that the tool works correctly, the A/B testing capability is developed, and can be easily incorporated with the development process.

1.1 Thesis objective and research questions

The objective of this thesis is to develop A/B testing capability for a development process using a third-party A/B testing tool. The process of developing the A/B testing capability is documented to provide guidance on how to enable the capability for similar development processes and projects. Furthermore, a quick guide on A/B testing is created to ensure that the team members have the required knowledge to conduct A/B testing after the capability has been developed. Following research questions support the process.

1. *What is the development process in use and how A/B testing with a third-party tool can be incorporated with that?*

Studying and defining the development process in use allows examining how the A/B testing capability can be incorporated. Furthermore, when choosing a tool for A/B testing it needs to support the technologies already in use. Thus, identifying the important technologies regarding choosing a tool for A/B testing is necessary. Section 3 focuses on answering this research question.

2. *What aspects should be considered when choosing a tool for the development process?*

Selecting a software tool for a project is not a novel task. It can be considered as a multi criteria decision making problem where a set of available options represented by multiple different attributes are evaluated and ranked (Jadhav & Sonar, 2009). Thus, by using available literature on software tool selection and knowledge on the development process in use, aspects that should be considered when choosing a tool for A/B testing are identified in Section 5.4.

3. *How tools for A/B testing differ from each other and what tool is the most suitable for the process?*

The aspects recognized by the previous research question can be used as a guide in the comparison process. By examining the identified aspects for each compared tool, the differences and similarities between tools can be recognized. This helps to select the most suitable tool for the development process in use. The comparisons are conducted in Sections 5.3 and 6.

4. *What steps are needed for implementing A/B testing in practice?*

The different steps for developing an A/B testing capability for a project with a third-party tool are identified. This knowledge is further used in an A/B testing experiment. Furthermore, by creating a concrete list of these steps, other projects and similar processes can use it as a guide to conduct A/B testing in practice. The identified steps are presented in Section 4.2.

5. *Is the chosen tool suitable to use for the A/B testing in the development process?*

To test whether the chosen tool can be used to conduct A/B testing as part of the development process, the tool is deployed and a simple A/B test experiment is run. While conducting the experiment with the tool, the smoothness of the deployment and testing process is assessed to analyse whether the tool is suitable to deploy as part of the development process. The suitability of the tool is evaluated and analysed in Section 7.

1.2 Structure of the thesis

In Section 2, A/B testing and its related concepts are presented and discussed to provide necessary background information for the reader on A/B testing. Section 3 studies the development process and organisation behind it to help understand the needs and constraints that are present in the development of the A/B testing capability. Furthermore, the product, which the A/B testing experiment is conducted with, is presented. Section 4 describes the methodology behind the A/B testing tool comparison and selection and presents the steps for conducting an A/B testing experiment. The assessment criteria and the tools chosen for the more extensive comparison are identified in Section 5. Based on the identified assessment criteria, the tool comparison to select the most suitable tool is conducted for three chosen tools Section 6. In Section 7, the process of developing A/B testing capability with the chosen tool by conducting an A/B testing experiment with the tool is presented. Section 8 discusses the obtained results and considerations regarding the future. Finally, the thesis is concluded in Section 9

2 A/B testing

A/B testing is a type of an online controlled experiment where users of a service are randomly assigned to control and treatment groups (Aijaz, Stuart, & Jewkes, 2018). The users in the treatment group are shown an updated version of the service while the users in the control group are shown the original version. In this setup, the users are considered as experimental units, and their responses and behaviour are studied to calculate previously defined metrics for each group (Larsen et al., 2023). The aim of A/B testing is to assess whether there is a difference in the user behaviour between the groups. Traditionally, the means of the metrics calculated for each group are compared to determine if there is a difference between the groups (Kohavi & Longbotham, 2023). If the A/B test has been properly conducted, the only difference between the two groups is the new feature, which allows associating any statistical differences between the groups to the feature (Aijaz et al., 2018). A high-level diagram of the process of A/B testing is presented in Figure 1.

2.1 Use cases

The shift to agile development and continuous integration and deployment allows companies to rapidly deliver valuable software with the use of online controlled experiments (Aijaz et al., 2018). Continuous experimentation can be considered as a part of continuous deployment and development operations (DevOps) practises (Quin, Weyns, Galster, & Silva, 2024) and it is already a norm in large and progressive software companies (Fabijan et al., 2017). For instance, companies such as Google, Netflix and Facebook constantly run experiments (Aijaz et al., 2018). Furthermore, due to the availability of A/B testing tools, the costs of experimentation have dropped, thus allowing companies of varied sizes to implement experimentation (Koning et al., 2022).

OCEs, including A/B testing, can be applied to multiple platforms and areas, e.g. desktop, mobile, front-end or back-end (Larsen et al., 2023). By developing the testing capability for multiple platforms, the design choices for each platform can be validated and optimised as the same design decisions may not work for multiple platforms (Martin, 2015). For instance, users may prefer different UI elements on mobile compared to web. Furthermore, the platform may affect test run times as for example mobile users may need longer periods of time to familiarise with a change or a new feature (Martin, 2015).

The most common application domains in experimentation are web, search engines and e-commerce (Quin et al., 2024). OCEs can be used to test minor changes in the UI such as different button colours or larger, more significant changes such as new products (Koning et al., 2022). Overall, A/B testing is most frequently used for testing algorithms, visual elements or workflows (Quin et al., 2024). The results of an A/B test can be used in multiple ways, for instance, in feature selection or roll-out plan, research question validation or bug detection (Quin et al., 2024).

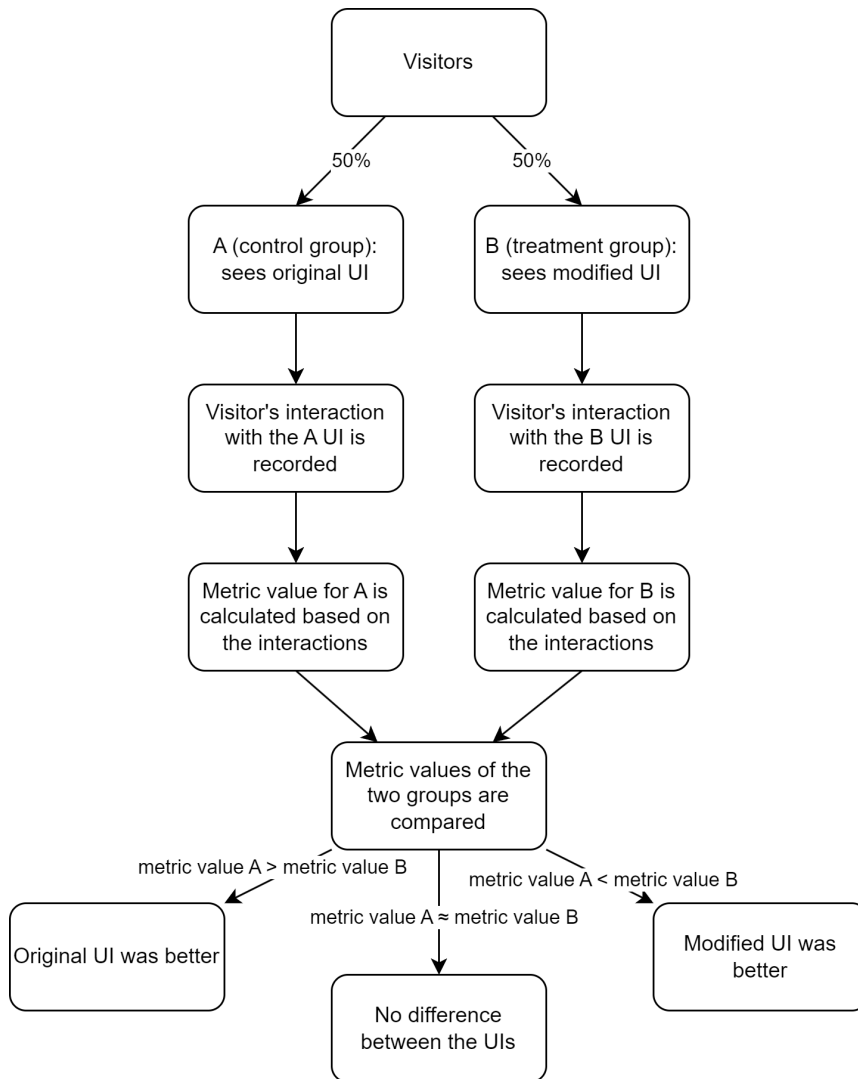


Figure 1: High-level diagram of A/B testing process.

2.2 Benefits and limitations

A/B testing can accelerate organisational learning by lowering the risk of new releases and allowing to execute changes faster (Koning et al., 2022). For instance, by incrementally experimenting and subdividing a new release or a change to small parts, its risk can be reduced (Koning et al., 2022). Furthermore, experimentation allows maximising revenue (Larsen et al., 2023).

When developing an A/B testing capability and adopting experimentation as part of the workflow, performance can improve from thirty to one hundred percent after a year (Koning et al., 2022). Additionally, A/B testing can ensure product quality, help to discover and validate value, and stabilise and lower the complexity of a product (Fabijan et al., 2017). Kohavi and Longbotham (2023) state that use of OCEs can help to detect small and unexpected changes that are hard to detect with other techniques. Furthermore, they claim that OCEs are the best scientific way to determine causality

with high probability.

Even though A/B testing or OCEs in general allow testing even profound ideas, they themselves are not a silver bullet for creating the next new radical design or product. Instead, they allow incremental and iterative testing and honing of design ideas (Kompella, 2015). However, OCEs must be conducted carefully to acquire accurate and useful results (Martin, 2015). Furthermore, even in large and leading companies only ten percent of the time the new features have a positive impact (Aijaz et al., 2018). And in general, only fifty percent of ideas result with meaningful improvements (Larsen et al., 2023).

Kohavi, Longbotham, Sommerfield, and Henne (2009) list additional limitations of online controlled experiments in their paper. For instance, A/B testing produces quantitative data which allows to determine the winning variant but does not elaborate the cause. Additionally, primacy and newness effects can cause biases to the data in A/B testing (Kohavi et al., 2009). For instance, moving UI elements from familiar places can initially cause users some trouble finding them which favours the control variant. Furthermore, when a new feature is implemented some of the users may investigate it causing bias towards the treatment variant. The effects can be minimised by running the experiment longer or by having only new users as part of the experiment (Kohavi et al., 2009).

2.3 Metrics

A/B testing metrics are used to calculate the difference between the control and treatment groups. Thus, they can be considered as fundamental factors in A/B testing (Machmouchi & Buscher, 2016). Because the metrics have such a big emphasis on the A/B testing and its results, it is important to understand how they should be constructed and managed. Furthermore, the metrics tend to easily become weak or faulty which leads to incorrect results; therefore, they need to be created carefully (Machmouchi & Buscher, 2016).

The metrics are based on events or observations such as button clicks or page load times and are calculated against an experimental unit. Traditionally, the user of the tested website is used as the experimental unit, however other experimental units such as sessions can also be used (Kohavi & Longbotham, 2023). Most common metrics are engagement metrics, for instance measuring the amount of conversation or how many times a user has completed specific action on the service (Quin et al., 2024). Second most popular metrics are click metrics (Quin et al., 2024). Metrics can also measure monetization, performance, or unwanted effects (Quin et al., 2024).

When developing an A/B testing capability, it is beneficial to create and monitor multiple metrics, for instance Kohavi and Longbotham (2023) recommend calculating hundreds of metrics from the observations if possible. These different metrics can be divided into groups or categories based on what they represent and how important they are.

The most important metric, which itself can be considered a metric category, is overall evaluation criteria (OEC) (Aijaz et al., 2018). The OEC is the metric which A/B testing aims to improve as it measures the long-term business value (Kohavi

& Longbotham, 2023). However, it should also measure the long-term customer satisfaction with the product (Fabijan et al., 2017). It attempts to depict whether the treatment group has expressed more desired behaviour compared to the control group (Machmouchi & Buscher, 2016). The OEC should be sensitive enough to detect changes in appropriate time periods and it should be directional (Aijaz et al., 2018). Kohavi and Longbotham (2023) suggest two weeks as an appropriate time to detect changes (Kohavi & Longbotham, 2023). Furthermore, the OEC should be easily understandable for the different stakeholders (Aijaz et al., 2018).

More often, even though a lot of effort and attention has gone into the design of the OEC, it is not perfect and can provide incorrect results in some situations (Kohavi & Longbotham, 2023). Therefore, while the decision to release the feature should be based on the OEC, other metrics should be created to support it (Kohavi & Longbotham, 2023).

While the OEC is created to measure long-term business value, there might be interest within the project to monitor and measure some aspects of specific features. For instance, if the location or visuals of a button is changed, a metric can be created to monitor whether the clicking amounts have increased. These metrics tracking specific features are called feature metrics (Aijaz et al., 2018).

When trying to improve the value of the OEC, it can lead to over optimization which can harm the service in other ways, for instance the page load times could increase. Therefore, creating guardrail metrics that monitor other aspects should be created so that the overall quality does not decrease when trying to improve the OEC (Aijaz et al., 2018). Such as the OEC, the guardrail metrics should be sensitive and directional (Aijaz et al., 2018).

2.4 Statistics

A/B tests are based on Null hypothesis testing, where the hypothesis presumes that the treatment has no effect on the calculated metric (Aijaz et al., 2018). The means of the metrics for control and treatment groups are calculated and compared to analyse whether the Null hypothesis can be invalidated. This means that there is statistically enough evidence that the difference between the means is caused by the effect of the treatment (Aijaz et al., 2018). Traditionally, statistical analysis in A/B tests is done using t-test together with p-value (Aijaz et al., 2018). The p-value represents the probability that the difference seen between the means is caused by sampling error (Kohavi & Longbotham, 2023).

There are two types of errors related to Null hypothesis testing. The type I error, i.e. false positive, occurs when the Null hypothesis is valid, but it is incorrectly considered as false (Aijaz et al., 2018). The probability of type I error represents the significance level α . The type II error, i.e. false negative, refers to a situation where the Null hypothesis is false, but it is incorrectly considered as true (Aijaz et al., 2018). The probability of type II error can be used to calculate the sensitivity, also known as the power, of the hypothesis: $1 - \beta$ where β represents type II error probability (Aijaz et al., 2018) (Kohavi & Longbotham, 2023). The power represents the ability of detecting an existing difference (Kohavi et al., 2009).

The probabilities can be lowered by increasing the sample size; however, it is standard practice to set them to fixed values: $\alpha = 0.05$ and $\beta = 0.2$ (Aijaz et al., 2018). If it is important that the experiment is not incorrectly declared as successful, the value of α can be set to 0.01 (Aijaz et al., 2018). The sensitivity of the experiment can be increased in different ways in addition to increasing the sample size e.g. by using a more sensitive metric or by transforming the metric (Kohavi & Longbotham, 2023). The metric can be transformed by removing outliers, capping the value to a maximum value or by using log transformations (Kohavi & Longbotham, 2023). Finally, the experimental units should be halved to equal sized groups to increase the sensitivity (Kohavi & Longbotham, 2023)

To confirm that the hypothesis can be rejected or accepted with enough statistical evidence, the p-value is compared to significance level α . If the p-value is equal or greater than the significance level, the Null hypothesis can be confidently rejected and vice versa (Aijaz et al., 2018). However, this statistical calculation assumes that the distribution follows normal distribution, thus for the results to be statistically correct, the experimental unit sample size must be at least in thousands (Kohavi & Longbotham, 2023)

Typically, the t-test can be applied in A/B testing because the experimental units and randomization units are the same. However, if the experimental unit used in the calculated metric differs from the randomization unit, other statistical methods need to be used for the comparison, such as bootstrapping or the delta method (Kohavi & Longbotham, 2023).

2.5 Testing tools

In recent years, increasingly more A/B testing tools have been developed making the testing process faster, easier, and less expensive (Siroker et al., 2013) (Koning et al., 2022). The tools offer the A/B testing capability for the users by offering randomization algorithm, assignment method and statistical engine (Kohavi & Longbotham, 2023) (Aijaz et al., 2018). The minimal requirements of the A/B testing tool are that the randomization algorithm together with the assignment method should not cause any performance bottlenecks and that the users are indeed randomly assigned to the groups, and once assigned they should remain in the assigned group (Aijaz et al., 2018)

Using third-party testing tools to implement A/B testing capability has other advantages besides not needing to design and create the tool by yourself. For instance, they eliminate the need to understand the statistics behind A/B testing (Kompella, 2015). The users of the tools can trust that statistical calculation and reporting are trustworthy and accurate as they are carefully developed by the offering companies (Siroker et al., 2013). However, Larsen et al. (2023) still suggest doing validation checks on the quality of the data to ensure that the tool is working properly. Furthermore, purchasing a tool gives access to professional support, consulting and a community support that helps with technical support and provide best practices, and the tools often contain many advanced testing features (Siroker et al., 2013). The following list contains some of the common features the tools may provide.

- *Visual editor*: Many of the tools offer visual editors or otherwise known as What You See Is What You Get (WYSIWYG) editors, allowing team members without programming background to create and tests variants ([Kompella, 2015](#)).
- *Multivariate testing*: Tools usually provide multivariate testing capability in addition to the A/B testing capability ([Kompella, 2015](#)). Multivariate testing is further discussed in Section 2.6.
- *Multipage testing*: Tools enable testing multi-step processes ([Kompella, 2015](#)) ([Martin, 2015](#)).
- *Segmentation*: A/B testing often allows conducting segmentation ([Kompella, 2015](#)) ([Martin, 2015](#)). In segmentation, after the tests are run the experimental units (users) are divided into isolating groups to analyse further the performance of each segment ([Siroker et al., 2013](#)). The different segments could be for instance web, mobile and tablet users.
- *Automation*: Tools automatically present the results of an experimentation and declare the best-performing variant ([Kompella, 2015](#)).
- *Mobile app A/B testing*: Support for mobile app testing can also be offered by the tool, enabling the use of the same tool for the web and the mobile A/B testing ([Kompella, 2015](#)). Furthermore, mobile app A/B testing can allow testing without submitting an updated version to the application store. This is a huge benefit as it can take days to get an application update approved for the application stores.

There are many tools available in the market such as Optimizely, Visual Website Optimizer (VWO), Adobe Target, A/B tasty and so on ([Koning et al., 2022](#)). Furthermore, feature flag tools such as Split offer A/B testing capabilities as well ([Split, 2024b](#)). Due to the volume of available tools, choosing one can be difficult. This problem of choosing a tool and what aspects should be considered in the selection process is discussed in more detail in Section 4. However, the following list contains few considerations that could be made when choosing an A/B testing tool.

- *Ease of use*: A tool that is easy to use allows non-technical team members to create and run tests, thus freeing the time of the development team ([Kompella, 2015](#)).
- *Integration with current technologies*: How easily the tool can be integrated with currently used tools, systems and development mechanisms is an important factor to consider when choosing a tool ([Kompella, 2015](#)) ([Siroker et al., 2013](#)). The tool should not require significant changes to the current processes and code base. Furthermore, if the tool is well integrated with the underlying systems, such as the analytics tool, it helps to create efficient and fine-grained experimentations ([Kompella, 2015](#)) ([Siroker et al., 2013](#)).

- *Client versus server*: The tools can be deployed either on the client or server side depending on the needs. For instance, client-side implementation can be easier but may cause performance problems, meanwhile server-side implementation can be more flexible (Kompella, 2015). Some tools provide both options, for instance Split offers both client and server-side software development kits (SDKs) (Split, 2019e).

2.6 Variants

A/B testing can be modified to different variants for different purposes. Testing with two identical groups is considered as an A/A testing or Null testing (Kohavi et al., 2009). In A/A testing the allocation of the users to the different groups is done, but they are both shown the same treatment (Kohavi & Longbotham, 2023) (Aijaz et al., 2018). It can be used to validate that the randomization algorithm, statistical engine, and overall setup is properly functioning (Kohavi & Longbotham, 2023) (Aijaz et al., 2018). Furthermore, A/A testing can be used for data collection to assess power calculation variability (Kohavi et al., 2009). A/B testing and tool validation using A/A testing is discussed more in Section 2.8.

A/B testing can also be expanded to multivariate testing, where multiple elements are simultaneously tested (Kompella, 2015) (Kohavi & Longbotham, 2023). It allows examining the relative impact of an element or combination of elements against studied metrics (Martin, 2015) (Kohavi et al., 2009). It allows faster testing e.g. testing five different elements simultaneously takes two weeks versus testing each new element individually for two weeks takes ten weeks. However, some combinations of elements can decrease the user experience or give poor results even if the variants alone improve the service (Kohavi et al., 2009). Furthermore, the result analysis can be more complex and the result interpretation difficult (Kohavi et al., 2009). In addition, beginning to run the test can take longer as all the variants must be ready versus in traditional A/B testing only one variant needs to be ready at a time (Kohavi et al., 2009). Kohavi et al. (2009) suggest beginning the online experimentation with A/B tests rather than multivariate tests due to their complexity.

2.7 Implementation

When developing A/B testing capability in a project, the firsts tests should be simple and small and gradually move to more complex tests (Martin, 2015). The starting point can be a landing page or other high-traffic page (Martin, 2015). Furthermore, literature provides following steps for the A/B testing implementation.

The first step is to define the purpose of the service and aspects that need improvement (Siroker et al., 2013). For instance, the purpose of the digital service of an electricity sales company can be to provide the customers means to view and access their electricity information. Furthermore, there may be a desire to guide more of the users to self-service functionalities.

Using the information on the goals and problems of the service, a metric or metrics are defined which allow tracking and assessing the user behaviour (Martin, 2015).

The point where a user takes a desired action on the service is called conversion and it can be tracked using metrics (Siroker et al., 2013). For example, a metric used to track whether there is improvement on guiding the users to use self-service functions, could be formulated as *average of self-service actions taken per user*. Furthermore, as discussed previously, it is good practice to create multiple metrics for the experiment (Kohavi & Longbotham, 2023).

Next step is to identify any bottlenecks currently present on the service (Siroker et al., 2013). These bottlenecks are the points where users are dropping off from the path to desired action. Analytics data or user studies can be used to identify these points (Siroker et al., 2013). For instance, a bottleneck could be that users do not recognize the availability of self-service functions and therefore contact the customer service via phone or email.

After identifying bottlenecks, a hypothesis is constructed (Siroker et al., 2013) (Martin, 2015). The hypothesis should be created using the knowledge of the user and it helps to make the tests more informative by providing a better comprehension of what the experiment tries to determine (Siroker et al., 2013). An example of a hypothesis could be *"The users are not aware that they can conduct self-service functionalities on the service; by displaying the information clearly the users should more frequently resort to self-service action than to contacting customer service"*.

Next, based on the hypothesis, different variations of the elements or the site are considered, and a new variation is created that should improve the metric based on the hypothesis (Siroker et al., 2013) (Martin, 2015). Then finally, the A/B test can be run (Siroker et al., 2013). The test should be running at least a full week to acquire a large enough sample size of both users and different times (Martin, 2015). Furthermore, if there is a suspicion that the initial effect of the treatment differs from the long-term effect, the test should be run until the long-term effect can be detected (Kohavi & Longbotham, 2023). After the test has run, it often results with more questions that require answers. Thus, leading back to the first step. Overall, the steps can be considered as a part of an iterative cycle (Siroker et al., 2013). Figure 2 displays these steps and how they form a cycle.

2.8 Validation

As mentioned before, A/A testing can be used to validate the A/B testing tool and setup. For instance, when using confidence level of ninety percent, ten percent of the time the Null hypothesis should be rejected, thus matching the defined acceptable type I error rate (Kohavi & Longbotham, 2023) (Aijaz et al., 2018). If this does not hold, there is an indication that the A/B testing setup does not work properly. To properly validate the setup, the A/A testing should be conducted multiple times because for instance when the used significance level α is 0.05, there is a five percent chance that a metric result is a false positive. Thus, if only run once and some metric displays a difference it might not be accurate but due to the type I error. If the A/A test results constantly indicate a difference between metrics the setup is not working correctly. This might be due to problems in either randomization algorithm, targeting engine or statistical engine (Aijaz et al., 2018).

OEC can also be validated. Validation is done by conducting an experiment with a poor feature that should decrease the value of the OEC. If the OEC does not degrade it is not working properly and thus it is not a good OEC metric (Aijaz et al., 2018).

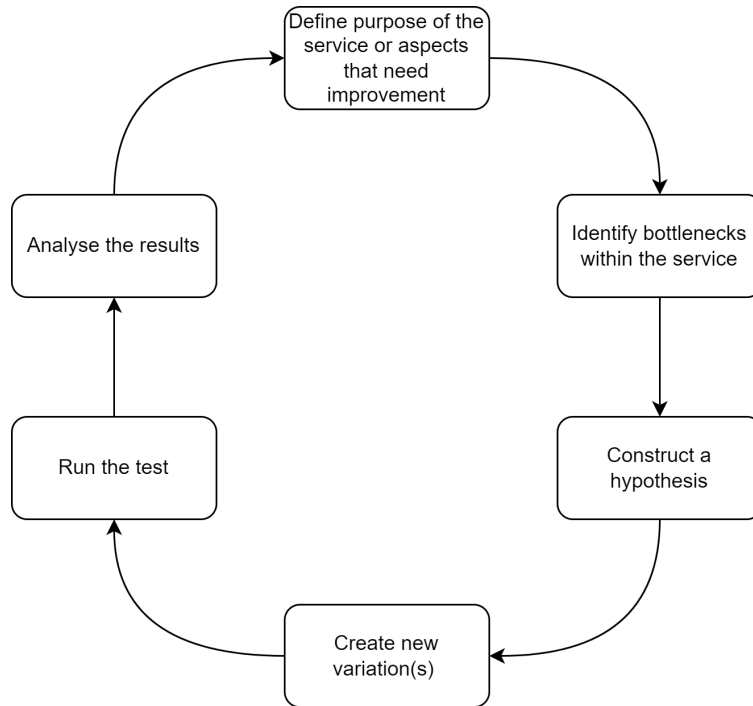


Figure 2: A/B testing implementation steps.

2.9 Ethical questions

There has been discussion about the ethics of online controlled experiments. The discussion has especially sparked due to Facebook’s emotional contagion experiments and OKCupid’s matching algorithm manipulation (Jiang, Martin, & Wilson, 2019). Even though these experiments go beyond the traditional A/B testing, it is still important to recognize and understand ethical questions related to online experimentation.

Traditionally, in academia when conducting user research, the participants are always asked for consent (Jiang et al., 2019). However, these online experimentations, such as A/B testing, disregard requesting the participants explicit consent (Benbunan-Fich, 2017) (Jiang et al., 2019). Furthermore, some companies ask for the consent with Terms of Service agreement, which is complex, long, and hard to comprehend by regular users therefore receiving informed consent is questionable (Benbunan-Fich, 2017). This leads to situations where the users are used in experiments without their knowledge. Thus, there should be some discussion on whether it is ethically acceptable to conduct the experiments on users without their consent and how the consent should be requested.

Another ethical concern regarding A/B testing or online experimentation is what these experiments contain, especially what is tested and how (Jiang et al., 2019). For

example, there are multiple instances where A/B testing has been used for price or advertising discrimination (Jiang et al., 2019). Furthermore, A/B testing can be used to affect political and voting behaviour which increases the political polarisation (Jiang et al., 2019). Overall, when conducting online experiments any changes that can harm visitors in any way, for instance, financially, socially, or psychologically can be seen as unethical (Kohavi & Longbotham, 2023).

Kohavi and Longbotham (2023) suggest asking two questions when conducting an online experiment (Kohavi & Longbotham, 2023). Firstly, would it be okay given the organisational standards to ship the change to all users, and secondly, if the experiment were published in nationwide media, such as newspapers, would it cause a public relations problem. By answering these questions, it should give an indication whether there are ethical problems related to the experiment.

3 Development process and product description

The objective of this thesis is to develop A/B testing capability for the development process used in an electricity sales company. The process is used to develop a digital service for the customers of the company. Studying the process and the product can help to identify assessment criteria for the tool. Furthermore, familiarising with the product helps to formulate the A/B testing experimentation.

3.1 Development process

There are multiple development teams in the organisation. However, many of the teams follow the same development process. Thus, by developing the A/B testing capability for one process and documenting it, the documentation can be used as a guide in teams developing other products with similar processes.

The process uses agile methodology and runs in two-week sprint cycles with predetermined ceremonies. These ceremonies include for instance, sprint planning, daily scrums, backlog refinement, sprint demo and sprint review. Furthermore, DevOps practices such as continuous integration (CI) and continuous deployment (CD) are used within the development process. The CI/CD pipeline can support for instance automated testing, integration, building and deployment. In addition to these, DevOps practices can include continuous experimentation which enables data-driven development (Quin et al., 2024). By developing the A/B testing capability for the process, continuous experimentation can be enabled (Quin et al., 2024).

The development teams using the development process should contain a product owner (PO), a Scrum master and actual developers. The developers may include software developers, designers, quality assurance engineers and DevOps engineers. The A/B testing capability should be achievable with the current team members.

Developing new features for the user interface (UI) has different steps. For instance, user feedback or the business unit may wish for a new feature. The PO determines whether the feature is taken to development. The UI of the new feature is designed by the designers and based on the design the developers implement the feature. The feature is then tested and approved for production.

3.2 Product

The product is a digital service targeted for the customers of the company. The service is offered on both web and mobile applications. With the service, the users can e.g. follow their electricity consumption and price, view their contracts and invoices, and handle some self-service tasks, such as updating their billing address or interval. The web platform has around four thousand daily active users while the mobile application has forty thousand. In this thesis the A/B testing capability is developed for the web application and more specifically for the front-end. However, while developing the A/B testing capability, the choices in tool selection should allow further expansion to server-side and mobile A/B testing.

3.2.1 Users

The users of the service are mainly electricity sales customers. The customer base is wide and there is no specific demographic portraying the users as anyone who needs to buy electricity in Finland can be a customer. Therefore, any assumption e.g. about the knowledge of the field or technical skills cannot be made when creating the experiment. Based on the data the users most commonly visit the landing page and the electricity page, which includes for instance their electricity usage information. The first A/B testing experiments should be done on either of these pages due to their high traffic.

3.2.2 Goals

The service provides insight cards on its landing page. These cards may include information on the current electricity spot prices, electricity consumption, or guide the user to take actions on the service. There is interest in learning how the position of the card related to other cards affects how much users interact with the card. By learning how much the position affects, it can be used to place the most important cards in places that the users most likely interact with.

Additionally, the service offers self-service functionalities, such as updating billing address, changing payment date, sharing access rights, transferring contract to new address, and path to renewing contract or changing its type. It is more cost-effective to allow users to complete these tasks independently online, than to have them contact traditional customer service. Thus, there is a long-term goal to decrease the contacts made to customer service by offering the users different self-service functionalities and guiding them to them. This long-term goal can be used as a base when developing an OEC.

3.2.3 Technologies and practices

The front-end of the service is written using TypeScript and React is used as a component library. There is also a design system available for the product development team. The design system creates more complex design components such as accordions or popover menus with correct styling. The design system library is also created using TypeScript and React. The back-end of the service uses Python as the programming language and Flutter is used to develop the mobile application for the service. Furthermore, Piwik PRO is used as the analytics tool for both web service and mobile application.

In the development and release process, feature flags are often used as tool. For instance, they allow developers to develop new features before they are meant to be released by hiding them. Furthermore, with the feature flags different service breaks can be done easily by flipping the flag from on to off. As feature flags allow displaying different versions of the interface, they can be utilised in A/B testing. Feature flagging capability for the front-end is achieved by using Split as the tool and the mobile application uses Google's Firebase as the tool. In the back-end there is no feature flagging tool in use.

3.2.4 Restrictions

The company is Finnish, and the product is targeted for customers of the company. Thus, regulations present in the EU must be considered. The most important regulation to consider when taking a new third-party tool to use or when handling user data is the General Data Protection Regulation (GDPR) ([European Parliament & Council of the European Union, 2016](#)) which needs to be followed. In addition, ePrivacy Directive concerning cookies should be considered ([European Parliament & Council of the European Union, 2002](#)). For instance, users cannot be tracked without their consent and therefore the tool should support conditional tracking. Overall, when taking new tools or practices into use, legal advice should be sought to ensure that all the regulations are fulfilled.

4 Methodology

There are two empirical phases in this thesis. Firstly, A/B testing tool is selected using a systematic comparison. Secondly, the selected tool is deployed, and A/B testing capability is developed and assessed by conducting A/B testing experiment using that tool. The methodology and processes used for the empirical phases are described in this section.

4.1 Tool comparison and selection

A comparison between different tools that enable A/B testing is conducted to evaluate what tools exist on the market and what differences they have. The objective of the comparison is to find a suitable tool for the development process and product presented in Section 3.

Selecting a tool for a software development process is becoming more challenging due to the increased number of different tools available (Bjarnason, Åberg, & Ali, 2023). To address this challenge of tool selection, ISO/IEC 20741:2017 standard presents a four-step process for evaluation and selection of software tools (Bjarnason et al., 2023). In addition, for example Bjarnason et al. (2023) and Jadhav and Sonar (2009) have presented their own models for tool selection process. Overall, there are multiple different methods and models to choose from when selecting a software tool (Bjarnason et al., 2023). In this thesis, the selection process is done by combining various aspects of these discussed models. This process is presented in Figure 3. The process contains two parts with multiple steps. The first part, where tools are chosen for in-depth comparison and assessment criteria is identified for the comparison, contains the first four steps. The last two steps comprises the second part and are focused on the comprehensive comparison and selecting the winning candidate.

In the first step, a list of possible A/B testing tools is created (Bjarnason et al., 2023) (Jadhav & Sonar, 2009). Then, common, and necessary functionalities are identified and used for initial comparison of the tools. Furthermore, the tools interoperability with the case project is examined, as it is an important aspect when selecting a tool (Jadhav & Sonar, 2009). In this case, the tool must support the programming languages used in the case project. Additionally, it would be beneficial that the tool has support for integration of the data analytics tool used in the case project and the tool provides feature flagging functionality.

In the third step, using the functional and interoperability comparison, some of the tools are eliminated. This elimination step corresponds to a step in the model presented by Jadhav and Sonar (2009). In the fourth step, assessment criteria for the comparison are identified using literature and knowledge on the development process, product, and A/B testing.

In the fifth step, the remaining tools are studied thoroughly and compared by using the assessment criteria (Bjarnason et al., 2023). The best suited tool is selected. The sixth step of the process is optional and conducted if there is no clear winner. In the step, analytic hierarchy process (AHP) is used to determine the best tool. The AHP is

used only if needed, as pairwise comparisons of criteria and tools are time consuming (Jadhav & Sonar, 2009).

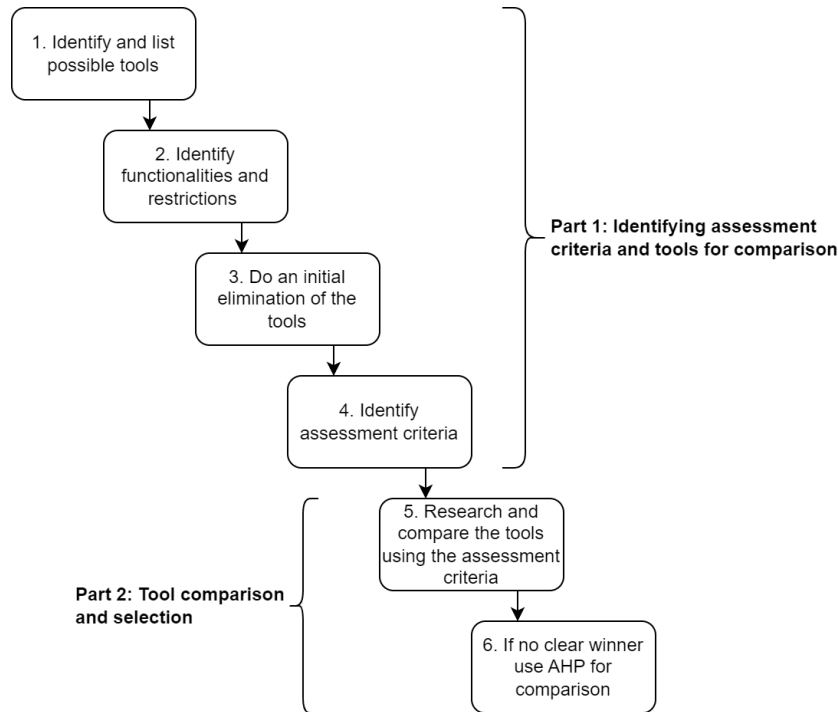


Figure 3: Tool selection steps.

4.2 A/B testing experiment

In the experiment phase, A/B testing capability is developed and evaluated with the product discussed in Section 3 by running A/A and A/B tests. After the experimentation is done, the team of the product should be able to conduct more A/B tests without large effort. The experimentation steps are based on the literature on implementing A/B tests, which was discussed in Section 2.7. In addition to those steps, some steps regarding the tool deployment and validation are added to the experimental phase. Furthermore, some preliminary actions are conducted in the experiment. The following list depicts all the high-level steps in the A/B testing experimentation.

1. *Preliminary actions:* Description of aspects that need to be considered and conducted before the tool can be taken into use and testing can be done. This includes for instance legal considerations and identification of metrics and the testable area.
2. *Tool deployment:* Taking the tool into use in the development project.
3. *Metric creation:* Creating relevant metrics concerning the testable area based on tracked events.

4. *Validation*: The tool is validated by running an A/A test.
5. *Constructing a hypothesis*: A testing hypothesis is constructed for the identified problem or testable area.
6. *Creating the treatment variant*: New variation based on the hypothesis is created to use as the treatment variant in the A/B testing
7. *Running the A/B test*: Running the A/B test for a whole week with the control and treatment variants. If there is not enough data gathered from one week of running, the test should be run for another week.
8. *Analysing results*: The A/B test results and experimental process is analysed and observations on the ease of use or potential problems with the tool are noted.

5 Identifying Assessment criteria and tools

First part of selecting an A/B testing tool for the development process is to identify the tools and assessment criteria for the comparison. Comparison is a time-consuming and laborious work, therefore only few tools should be thoroughly compared. This section describes the process of determining the few most suitable tools for the comparison. Furthermore, the assessment criteria used in the second part of the comparison are identified based on literature, the development process in use, and the focus area of the tool.

5.1 Selecting an initial list of tools

The A/B testing tools for initial comparison need to be selected. To select the most relevant tools, the top three tools with highest market share, and the top three tools from Gartner's list are chosen. The top three in highest market share are Optimizely, Adobe Target and VWO ([Datanyze, 2024](#)). And the top three in Gartner's list with the highest number of ratings are VWO, Adobe Target and AB Tasty ([Gartner, 2024](#)). All these tools were also listed by [Koning et al. \(2022\)](#) in their paper.

In addition, the project organisation is interested in studying the possibility to expand currently used feature flag tools into A/B testing tools. The tool used in web front-end development is Split. Google's Firebase is used in mobile development for handling feature flags. Both tools claim to provide A/B testing capabilities on their sites ([Split, 2024a](#)) ([Firebase, 2024a](#)), thus they are also chosen for the comparison. Thus, the initial list of A/B testing tools are Optimizely ([Optimizely, 2024](#)), Adobe Target ([Adobe for Business, 2024](#)), VWO ([VWO, 2024a](#)), AB Tasty ([AB Tasty, 2024c](#)), Split ([Split, 2024b](#)), and Firebase ([Firebase, 2024b](#)).

5.2 Identifying functionalities and restrictions

The common and necessary functionalities of the A/B testing tools and interoperability restrictions need to be identified to eliminate some of the selected tools.

The common functionalities of A/B testing tools presented in Section 2.5 is used as the initial list. Furthermore, as described in Section 3 the development process of the product relies on using feature flags. By selecting a tool which has feature flag and A/B testing functionality, the development and maintenance work can be minimised. In addition, the organisation can save money by using a single tool. Thus, feature flagging can be considered as a necessary functionality for the tool. While investigating the different tools, many of them marketed targeting and server-side testing. Thus, those are also included in the functionalities.

The tool must be interoperable with the current technologies used in the development process for the experimentation to be possible. Especially important is that the tool supports the programming languages used in the development process of the product. In future, it should be possible to use the tool for A/B testing in back-end and mobile development as well. Thus, the tool must support all the programming languages used in the project which are JavaScript/TypeScript, Python and Flutter. In addition, it

would be beneficial for the tool to support integration with the analytics tool in use, PiwikPRO, as it can allow exporting already tracked events by the analytics tool to the A/B testing tool.

5.3 Initial comparison and elimination

The information for the initial comparison is gathered from the websites and documentation documents the different tools provide. Based on the found information, two tables can be constructed. Table 1 displays the functionalities offered by different tools. Table 2 displays the interoperability of the different tools with the development project technologies. In addition to information found on the tools' sites, list of compatible tools provided by the data analytic tool, Piwik PRO, is used to create Table 2 (Piwik PRO, 2024).

Based on the information provided on their sites, tables comparing the identified functionalities and the interoperability of the tools with the development process of the product can be made.

Table 1: Included functionalities of Optimizely (Optimizely, 2024), Adobe Target (Adobe for Business, 2024), VWO (VWO, 2024a), AB Tasty (AB Tasty, 2024c), Split (Split, 2024b), and Firebase (Firebase, 2024b).

	Optimizely	Adobe Target	VWO	AB Tasty	Split	Firebase
Visual Editor	x	x	x	x		
Multivariate Testing	x	x	x	x	x	
Multipage Testing	x	x	x	x		x
Segmentation	x	x	x	x	x	x
Targeting	x	x	x	x	x	x
Automation	x	x	x	x	x	x
Feature Flags	x	x	x	x	x	x
Server-side Testing	x	x	x	x	x	
Mobile App A/B Testing	x	x	x	x	x	x

Some observations can be made from Table 1. Notably, Firebase offers the least amount of the listed functionalities, however it is still in beta phase (Firebase, 2024a). Most likely at some point the other features are also added to the tool. Split offers the second least number of features as it does not have a visual editor, nor does it directly advertise multi page testing capabilities. However, almost all the features are offered by all the different tools. Thus, when choosing an A/B testing tool, it can be expected that the different tools provide similar basic functionality. Therefore, to help with tool selection, other aspects must also be considered.

Table 2: Interoperability of the different tools.

	Optimizely	Adobe Target	VWO	AB Tasty	Split	Firebase
JavaScript / TypeScript	x	x	x	x	x	
Python	x	x	x	x	x	
Flutter	x		x	x	x	
Piwik PRO Integration	x		x	x		

Based on Table 2, Firebase and Adobe Target can be eliminated as they do not support all the three languages used in the project. Additionally, VWO is marked to support Flutter, however the product with Flutter support does not have feature flagging functionality (VWO, 2024b) (VWO, 2024c) and it is eliminated. Split does not support integration with Piwik PRO. However, it is already used in the project as a feature flagging tool. Thus, expanding it to handle A/B testing would require the least amount of development work and therefore it is not eliminated. Thus, the chosen tools for more detailed study and comparison are AB Tasty, Optimizely, and Split.

5.4 Identifying assessment criteria

In the second part of the tool comparison and selection, the tools are introduced and assessed based on assessment criteria. Jadhav and Sonar (2009) have identified a large amount of assessment criteria applicable when choosing a tool for a software development project. From this list some suitable criteria are used as a base for the assessment criteria (Jadhav & Sonar, 2009). Furthermore, as the task is to choose an A/B testing tool, some criteria specific to A/B testing and the development process used are identified and utilised for assessment. The following list contains the different assessment criteria and their explanation:

- *Help documentation*

Help documentation is an important aspect when considering a tool and e.g. availability of a user manual and a troubleshooting guide should be considered when selecting a tool (Jadhav & Sonar, 2009). In addition, things such as best practices, checklists, frequently asked questions (FAQs) and video tutorials can help with implementation of the tool. Furthermore, the navigation of the documentation should be easy and efficient.

- *Data visualisation & analytics*

One main advantage of using a ready-made tool for A/B testing is their built-in processing of the data (Kompella, 2015). The tools provide analytics for the experiments, which allows to evaluate whether the experiment worked and if either of the candidates were better without additional work. Thus, examining what calculations the tools make and how they present the data is studied.

Jadhav and Sonar (2009) also list data visualisation as an assessment criterion in their paper. Furthermore, the tool should also support exporting the results in different formats to allow sharing them easily within the organisation.

- *Metrics*

Metrics are a crucial part of the A/B testing as the test results are based on the values of the metrics. Thus, the metric capability of the tool should be high, meaning that it should allow tracking various kinds of metrics.

- *Events*

The metrics are calculated based on events. Thus, it is important to know whether there are limitations to the events that can be tracked. Furthermore, how the tracking of the events is implemented into the code can have a large effect on how easy the tool deployment and usage is.

- *Running an A/B test*

Running an A/B test with the tool should be effortless and the testing process should easily integrate with the development process in use, i.e. running an A/B test should not cause a large overhead, as it discourages the team members from running the tests.

- *Effects on performance*

When A/B testing in the front-end is conducted, there must be some logic which determines whether there is an ongoing experiment in the user's view; is the user part of the experiment and what view the user should be shown. This must be implemented on the tool side correctly, so that no significant delay is visible for the user. Thus, it is important to know what performance effects the tools claim to have.

- *GDPR Compliance & consent*

As the A/B testing uses data gathered from users, it is extremely important that the data is handled properly. There are laws and regulations which need to be adhered to e.g. the GDPR in the EU ([European Parliament & Council of the European Union, 2016](#)). Furthermore, events should not be tracked and sent if the users have not consented to being tracked and used for the research ([European Parliament & Council of the European Union, 2002](#)). Thus, having a built-in method to manage consent and disable event tracking would be beneficial.

6 Tool comparison and selection

In the second part of selecting an A/B testing tool for the process the tools selected in the first part are researched and compared using the assessment criteria as a guide. These tools are AB Tasty, Optimizely and Split. Aim of the comparison is to select the most suitable A/B testing tool for the development process.

6.1 AB Tasty

AB Tasty is an optimization and experimentation platform originating from France (AB Tasty, n.d.-j). They offer six products: Web Experimentation (AB Tasty, n.d.-i), Feature Experimentation (AB Tasty, n.d.-b), Content Personalization (AB Tasty, n.d.-a), Rollouts (AB Tasty, n.d.-c), Search (AB Tasty, n.d.-h), and Recommendations (AB Tasty, n.d.-k). From these, the Feature Experimentations is the most relevant for the case project as it provides feature flagging as a service, allows experimentation with A/B testing, and supports web, mobile and IoT (AB Tasty, n.d.-b). The Rollouts product does not market A/B testing and the Web Experimentation does not support mobile and does not market feature flagging as a feature. The rest of the products do not provide either feature flagging service nor A/B testing capability. Thus, the tools research and comparison are conducted with the Feature Experimentation product.

6.1.1 Help documentation

AB Tasty has separated the developer documentation (AB Tasty, n.d.-d) and user manual (AB Tasty, n.d.-f) to their own separate pages. In the user manual they present concrete examples for some of the features such as A/B testing (AB Tasty, 2021a). The developer documentation has a guide for all the different SDKs and the application programming interface (API). They offer video tutorials for A/B testing, feature flagging and progressive deployment (AB Tasty, n.d.-g). Furthermore, AB Tasty provides a demo and a sandbox to help familiarise with the product (AB Tasty, 2022). However, there is no troubleshooting guide, and the frequently asked questions (FAQ) page has only two questions (AB Tasty, n.d.-e).

In their user manual, AB Tasty provides a four-step quick start guide to begin using the Feature Experimentation tool (AB Tasty, 2021d). Firstly, decision API or SDK need to be set up. The API is a representational state transfer (REST) API, and it is designed as language agnostic. Second step is to define flags and third step is to define key performance indicators (KPIs) i.e. metrics. Finally, user context keys must be set up to allow assigning visitors to a feature.

6.1.2 Events & Metrics

Events are sent either by making a POST request to the AB Tasty's API (AB Tasty, 2024b) or by using a visitor instance method which is provided in the SDK (AB Tasty, 2024a). When creating a visitor instance, it requires visitor identification such as `user_id`. Each event is associated with the following data: event category, name, and

description (AB Tasty, 2024b). Event category is either user engagement or action tracking. Furthermore, optionally a non-negative integer value can be assigned to an event.

AB Tasty Feature Experimentation supports four types of metrics: transaction, event, page view (on web or server), and screen view (on app or server) (AB Tasty, 2021b). For measuring the variation performance, one primary KPI (metric) can be set. In addition to the primary KPI, secondary KPIs can be set (AB Tasty, 2021b). There is no specification in the documentation on how many additional KPIs can be set.

6.1.3 Running an A/B test

With A/B Tasty Feature Experimentation, configuring an A/B test begins with selecting an A/B test template from their application dashboard (AB Tasty, 2021a). The A/B test is given a name and a description and primary and secondary KPIs are chosen. The different variations are defined in the service by assigning flags and their values to them. Flag's value type can be either text, number, Boolean, array, or object (AB Tasty, 2021a). Finally, allocation is set to either by manually or set to dynamic, meaning that the traffic is diverted to best performing variation automatically (AB Tasty, 2021a). Running the A/B test requires that some events are tracked and KPIs created and that the flags are taken into use in the project.

The A/B testing process with A/B tasty uses feature flags as the base of the tests, which supports the current development and release process. However, as there is a need to select and create an A/B test template every time a test is run, it may cause overhead.

6.1.4 Data visualisation & analytics

AB Tasty Feature Experimentation provides a reporting layout for visualising A/B test results. Results for each metric can be viewed and for each variation the number of unique visitors and conversions are displayed (AB Tasty, 2020). For transaction and conversion rate, AB Tasty calculates the improvement between the treatment variant against the control variant using Bayesian algorithm (AB Tasty, 2020). Additionally, for each treatment variant a percentage of the chances of winning is displayed and colour coded; if 95% or greater it is green, if between 5 to 95 percent it is orange and if less than 5 it is red (AB Tasty, 2021f). Furthermore, the reporting layout has a *reliable* status, if there is enough significant data collected for the results to be reliable (AB Tasty, 2021e).

The results are shown in real-time for the first twelve hours, and after that the results are updated in varying time intervals (AB Tasty, 2020). The reporting data can be exported in comma-separated values (CSV) format (AB Tasty, 2021c).

6.1.5 Effects on performance

AB Tasty claims that A/B tests can be run with the Feature Experimentation product without affecting the performance (AB Tasty, n.d.-b). Furthermore, they declare that the response time for Decision API is less than 50 milliseconds and response time for bucketing file is less than 5 milliseconds.

6.1.6 GDPR compliance & consent

Feature Experimentation product of AB Tasty is GDPR compliant (AB Tasty, n.d.-b). Visitor consent can be managed using the SDK (AB Tasty, 2024a). If the consent is set to `false`, the data collection features for that user are deactivated.

6.2 Optimizely

Optimizely is one of the most popular A/B testing tools, having the largest market share of A/B testing tools according to Datanyze (Datanyze, 2024). Furthermore, the tool is referred to in literature (Jiang et al., 2019) (Kompella, 2015). The company is US based and offers multiple different plans for orchestrating, experimenting, or monetizing (Optimizely, n.d.-e). Web Experimentation and Feature Experimentation plans allow A/B testing. From these two, the Feature Experimentation plan is most suitable for the development process, as it provides support for mobile and server testing as well (Optimizely - Dev guide, 2024). Thus, that plan is researched and compared in this thesis.

6.2.1 Help documentation

Optimizely provides a large user and developer documentation (Optimizely, n.d.-c) (Optimizely, n.d.-d). They provide for instance FAQs (Optimizely, n.d.-b) (Optimizely, 2024b), troubleshooting guides, (Optimizely, n.d.-f) (Optimizely, 2023e) and an SDK implementation checklist (Optimizely, 2023c) to help with the implementation process and tool usage. Furthermore, they also provide videos to overview the product (Optimizely - Dev guide, 2024). The developer documentation contains a code example for each SDK which demonstrates how to evaluate feature flags and run A/B tests (Optimizely, 2024a).

In addition, Optimizely lists four steps for getting started in using the Feature Experimentation tool (Optimizely - Dev guide, 2024). First step is to install and initiate the SDK or to use JavaScript Browser HTML script quick-start method. Second step is to define feature flags. Third step is using the flags to run the A/B test and the last step is to analyse the results. They also provide a quick start guide for each Feature Experimentation SDK (Optimizely, 2024b).

6.2.2 Events & Metrics

In Optimizely Feature Experimentation events are sent using SDK's user context instance method (Optimizely, 2023a). Creating the user context requires using an

identification for the user such as `user_id`. Each event has a key which identifies it (Optimizely, 2023d). Additionally, events can be associated with optional tags, which is a map of key-value pairs where the accepted value types are string, Boolean, and number. Optimizely Feature Experimentation has support for unique conversions, total conversions, and total value metric types (Optimizely, 2024e). You can set multiple metrics but there is no specification on how many (Optimizely, 2024c).

6.2.3 Running an A/B test

With Optimizely Feature Experimentation the A/B tests are created on their application. They provide an example of how to run an A/B test on their SDK quick starts (Optimizely, 2023b). They also provide additional guidance on running A/B tests (Optimizely, 2024c). Before starting an A/B test it is required that a user context is created and used to track some event. In addition, a feature flag and at least two variations (control and treatment) need to be created on the application and implemented in the code. On the application, A/B test rule is added to the feature flag and a new event is created for the rule. The event should be the same as tracked in the code. Then, in the application a metric for the A/B test is created using the defined event as the event. Finally, the variations are set to match the desired variations and the test is saved. Then the A/B test can be enabled and run.

With Optimizely, there seems to be a lot of overhead when running an A/B test. In addition to requiring a rule to be set to a feature flag on their service, an event and metrics must be created for the test every time. Overall, there are many steps to run an A/B test. Thus, even though the testing is based on feature flags, running a test requires a substantial amount of effort.

6.2.4 Data visualisation & analytics

Results for the experimentations, including A/B tests, are presented in their application on the result page. The page contains general information such as number of visitors, days running, and experiment health indicating whether there are problems with sample ratios (Optimizely, 2024a). Furthermore, the page presents a summary of the experiment giving a high-level overview of the experiment (Optimizely, 2024a). Furthermore, there is a metric module for each metric containing information on unique conversions and total conversions, conversion and improvement rate, confidence interval, and statistical significance (Optimizely, 2024a). Optimizely also allows viewing improvement, visitors, conversions, conversion rate and statistical significance over time in a graph format (Optimizely, 2024a). There is no specification on what statistical methods Optimizely uses for calculating the results.

The results are shown in real time (Optimizely - Dev guide, 2024). Additionally, Optimizely Feature Experimentation provides a stat accelerator feature, which allows obtaining statistically significant results faster by manipulating traffic using machine learning (Optimizely, 2024d). The stat accelerator needs to be enabled when creating the experimentation to use it. Optimizely supports exporting the result data in CSV format (Optimizely, 2024a).

6.2.5 Effects on performance

Optimizely claims low latency for their Feature Experimentation tool by having in-memory bucketing ([Optimizely - Dev guide, 2024](#)). They promise that decision time is under a millisecond as there is no need to make any blocking API requests to receive decision ([Optimizely, 2024b](#)).

6.2.6 GDPR compliance & consent

Optimizely is GDPR compliant ([Optimizely, n.d.-a](#)). However, they do not offer a built-in consent method in the Feature Experimentation plan.

6.3 Split

Split is a US based Feature Delivery Platform which was founded in 2015 ([Split, n.d.-a](#)). Rather than offering various products they offer different plans with increasing amounts of features. The plans start from developer, which is a free plan and move on to team, business, and finally to enterprise ([Split, n.d.-b](#)). From business and onward, the plans offer A/B testing capability.

6.3.1 Help documentation

The developer documentation and user manual are located in the same site ([Split, 2018a](#)). API documentation is located on a separate site ([Split, 2023](#)). Split offers comprehensive documentation. For instance, for the following sections: Installing the SDK, Setting up your organisation, Feature flagging & configuration, Monitoring & experimentation, and Integrate & automate, they provide best practices, FAQs, and video walkthroughs. For the SDK installation and usage, they provide a checklist ([Lynch, 2023](#)) and a troubleshooting guide ([Split, 2019f](#)). In addition, they offer documentation for each SDK and they provide examples using the different SDKs ([Split, n.d.-d](#)) ([Split, n.d.-e](#)). Split also provides guidance on overall experimentation, for instance how to construct a hypothesis ([Split, 2021b](#)).

Split offers a four-step guide for beginning to use the tool ([Split, n.d.-f](#)). First step is to install the SDK, after that a feature flag and target users can be created. Third step is to send event data and the final step is to create a metric to monitor and measure impact.

6.3.2 Events & Metrics

In Split, events can be sent either by using a built-in method provided by the SDK or by making a POST request to the Split API ([Split, 2018c](#)). When using the SDK method, the SDK needs to be instantiated which requires setting a key which represents the `user_id` or if tracking anonymous users, a `cookie_id` ([Split, 2018d](#)). The needed parameters for sending an event with the SDK method differs between the SDKs. However, the event calls use the following information: key, which connects the event to a treatment (feature flag), traffic type (users, anonymous and so forth), and event

type which identifies the event (Split, 2018c). Optionally, a value (integer or float) or properties (key-value pair map) can be associated with an event (Split, 2018c).

Split supports five types of metrics: count of events per traffic type, sum of event values per traffic type, average of event values per traffic type, ratio of two events per traffic type, and percent of unique traffic type (Lynch, 2022b). Furthermore, when setting up a metric the desired impact needs to be specified (Lynch, 2022b). The desired impact can either be set to increase or decrease. The values or properties specified in an event can be used to calculate metric values. Metric values can be set to a cap where the outlier values are changed to the capped value (Lynch, 2022b). Split allows defining multiple metrics but there is no specification on how many.

6.3.3 Running an A/B test

There is no direct guide on how to run A/B tests with Split. However, they provide guidance on how to run an A/A test (Split, 2019d). Based on it, the steps for running an A/B test can be identified. Before running the test, the application should use Split SDK, events that support the desired metric should be sent to Split and the desired metrics should be created (Split, 2019d). Then, on their application a feature flag with default rule specifying a percentage-based rollout with fifty-fifty split should be created. Finally, the method for making the feature flag decision needs to be incorporated in the code (Split, 2019d).

A/B testing with Split is also based on feature flags. Thus, it supports the development process in use. In addition, once a metric is created in Split, it is being calculated for all the feature flags sharing the same traffic type even if there is no A/B test running (Ball, 2023). Thus, it allows to continuously follow whether flags affect the metrics. Furthermore, as there is no need to select or create metrics for each A/B test, the overhead is smaller.

6.3.4 Data visualisation & analytics

In Split the results are associated with the feature flags. For each flag a metrics impact page is presented which presents traffic over time for each variation, unique keys and amount of impressions for each metric, total keys and impressions, whether test has been run long enough to experience seasonality, duration of the experimentation, whether sample ratio is valid, the set significance threshold, amount of metrics with desired impact, undesired impact and inconclusive results, and finally for each metric (starting from the key metrics) a card is shown which contains the impact of the metric and whether it was desired, undesired or inconclusive (Split, 2018e). Additional information for each metric is shown by clicking a metric card. The additional information for each metric shows the data for each variant, which includes e.g. p-value, error margin, impact for each variant, total standard deviation, and 95th percentile (Split, 2019b). In addition, the values of minimum, median, maximum data-points are shown when possible to calculate.

Split uses a frequentist framework and offers sequential and fixed horizon testing methods (Split, 2020a). Split claims that with the sequential method, it is possible to

get results almost immediately, thus speeding up the experimentation process ([Split, 2020a](#)).

Split automatically calculates results in varying time intervals, starting from after 5 minutes of running the experimentation ([Split, 2018e](#)). It is also possible to recalculate the results manually. Split allows exporting the results in CSV, JavaScript Object Notation (JSON), and Portable Document Format (PDF) ([Split, 2021c](#)).

6.3.5 Effects on performance

Split stores the rollout plans locally with the SDKs, thus eliminating the need to make network calls for feature flags ([Split, 2019e](#)). In addition, Split claims to be able to serve feature flags anywhere in the world in less than 200ms due to their dual-layer content delivery network (CDN) architecture ([Split, 2019e](#)).

6.3.6 GDPR compliance & consent

Split is GDPR compliant ([Split, n.d.-g](#)). In addition, there is built-in parameter in the SDKs for user consent, which allows disabling the tracking and sending impression and event data to Split's cloud ([Split, 2018d](#)).

6.4 Comparison

All three tools, AB Tasty, Optimizely and Split, offer similar basic functionalities based on their documentation. There are some small differences regarding the selection criteria determined in Section 4. These differences are presented and discussed in this subsection, and they are used to determine the most suitable tool.

6.4.1 Help documentation

Table 3 showcases the differences in offered help documentation types. It is noticeable that the documentation by AB Tasty is not as extensive as the documentation provided by Optimizely and Split. Furthermore, the navigation of the documentation by AB Tasty is impractical as there is no side menu containing all the sections. As a result, Optimizely and Split are superior to AB Tasty, when considering the help documentation criteria. Optimizely and Split both offers extensive documentation and a winner between them cannot be declared.

6.4.2 Events & Metrics

Properties of the tools related to the assessment criteria of metrics and events are presented in Table 4. All three tools offer built-in SDK methods for tracking the events. Furthermore, AB Tasty and Split also offer tracking the events by making API requests. Split allows associating the event with either a number value or a map of key-value pairs, while Optimizely only supports setting a map of key-value pairs and AB Tasty only a non-negative integer value. Furthermore, Split has the widest support for different metric types and is the only tool from the three tools to offer metric

value capping. In conclusion, the Split seems to rank the best when considering the events and metrics selection criteria. Furthermore, Optimizely seems slightly better than AB Tasty by offering more metric types and by allowing more complex optional information associated with the events.

Table 3: Help documentation features.

	AB Tasty	Optimizely	Split
Troubleshooting guide	no	yes	yes
FAQs	limited	yes	yes
Checklist	no	yes	yes
Examples	demo and sandbox	code example for each SDK with feature flag evaluation and how to run an A/B test	repositories containing example projects using different SDKs

Table 4: Event and metric features.

	AB Tasty	Optimizely	Split
Event tracking	API request or SDK method	SDK method	API request or SDK method
Optional information associated with the events	value: non-negative integer	tags: map of key-value pair	value: integer or float, properties: map of key-value pair
Metric types	transaction, event, page view, screen view (not web)	unique conversions, total conversions, total value	count of events per [traffic type], sum of events values per [traffic type], average of the event values per [traffic type], ratio of two events per [traffic type] and percent of unique [traffic type]
Metric value capping	no	no	yes

6.4.3 Running an A/B test

All the tools used feature flags as their base for the A/B test, which suits the development process currently in use for the product. However, from the three tools, Split calculates the metrics for all the flags despite whether there is an A/B test in place or not. Furthermore, Split required the least number of steps when running an A/B test. Thus, it creates the least amount of overhead. When comparing the A/B Tasty and

Optimizely, AB Tasty has fewer steps as it does not require creating events or metrics but asks to select one or multiple KPIs (metrics). Thus, A/B Tasty has less overhead than Optimizely.

6.4.4 Data visualisation & analytics

All three tools provide comprehensive result pages which clearly demonstrate the winning candidate if there is any. From the three tools, Split seems to offer the largest amount of information on the results. However, Optimizely also provides wide range of result data.

For comparison, various aspects related to data visualisation and analytics are presented in Table 5. Again, AB Tasty has the fewest number of properties of the three tools. Optimizely and Split have similar features related to data visualisation and analytics. However, Split surpasses Optimizely on the amount of different export formats. Overall, Split and Optimizely offer similar features, with Split being slightly superior.

Table 5: Data visualisation & analytics features.

	AB Tasty	Optimizely	Split
Statistical methods	Bayesian	Not specified	Frequentist
Result acceleration	no	yes	yes
Sample ratio mismatch detection	no	yes	yes
Real-time results	first twelve hours	yes	no, but recalculation on demand
Supported result export formats	CSV	CSV	CSV, JSON, PDF

6.4.5 Effects on performance

The performance effects must be compared based on the claims from the different tools. AB Tasty claims that the tool will not affect the performance and their response time for decision API is less than 50ms while response time for bucketing file is less than 5ms. Optimizely also claims that there is low latency, due to in-memory bucketing, leading to decision making response time less than 1ms. Furthermore, Split claims to be fast by storing roll-out plans locally. In addition, by using dual-layer CDN architecture Split claims to be able to deliver feature flags anywhere in less than 200ms. From these three tools, Optimizely claims to be the fastest.

6.4.6 GDPR compliance & consent

All three tools are GDPR compliant. However, only AB Tasty and Split have a built-in SDK solution for handling the user consent and disabling the event tracking.

6.5 Results

Based on the comparison conducted in this section, AB Tasty seems to be inferior compared to Optimizely and Split. There are no single assessment criteria, where AB Tasty is solely superior, thus it can be ruled out.

Optimizely and Split both offer similar more advanced features, such as sample ratio mismatch detection and result accelerator. However, Split has an advantage on some aspects, e.g. by supporting a larger amount of metric types and having a metric value capping feature. Overall, Split seems to be slightly better than Optimizely, nevertheless the differences are quite minimal. With the suitability for the development process in use and the amount of overhead added to the process, Split is superior compared to Optimizely as it automatically tracks the metrics for each flag and there are fewer steps to run an A/B test. In addition, as Split is already used for feature flagging in the project, there exists an established practice in the development process for creating feature flags and therefore making the deployment of the A/B testing easier and faster. Thus, due to the comparison result and being already in use in the project, Split is the best tool for the development process and product described in Section 3.

7 A/B testing experiment

The development and implementation of the A/B testing capability is done by deploying the tool and conducting an A/B testing experiment. This section displays the experiment process following the outline of the steps presented in Section 4.2. Furthermore, observations on the use of the tool are noted and discussed in this section.

7.1 Preliminary actions

Before taking a tool or even a new feature of a tool into use the legal team should be contacted to ensure that all is according to the regulations and the tool or feature is safe to use. Furthermore, the goals of the testing and metrics for tracking them should be identified before beginning the testing process (Siroker et al., 2013).

7.1.1 Legal approval

When taking a new third-party tool into use, especially if it uses user data and/or its servers are in the USA, it should be screened and approved by the legal department of the company to ensure it is safe to use. In this case, as Split has already been used within the company, it had already been screened and approved. However, the tool was previously used only for feature flagging but not for A/B testing. Thus, the A/B testing features need to be investigated and approved by the legal team. This is especially important as the A/B testing process gathers data from actions of users and the tool and its servers are in the USA. The result from the legal team in this case was that the tool is okay to use as long as tracking and sending the data is only enabled if the user has accepted a cookie describing usage of the tool and data.

7.1.2 Modifying cookies

Based on the output from the legal team, the cookie consent texts need to be modified so that the user is aware of what data is collected and what it is used for. The cookie category needs to be determined. In this case, A/B testing seems to fall under the category of performance cookies. The decision was made together with multiple people with knowledge on the development field and cookies. In the organisation a third-party company manages the cookies and their content. Thus, the modification and deployment of the performance cookie text is made by personnel from the third-party company.

7.1.3 Identifying metrics and testable area

In Section 3 the goals and interests of the organisation was discussed. One interest within the organisation was to discover how much the location of a landing page card affects its clicking rate. This the landing page, and more specifically the card grid on the landing page, is considered as the testable area in this A/B testing experiment.

The knowledge on effects of card location can further be used to rearrange cards in a way that the cards that the organisation wants the users to notice are placed on

the locations which attract the most attention from the users. To study these effects, there should be metrics tracking the activity of the cards. Clicking rates can be used to measure activity on a card. Thus, the following metrics can be identified:

- *Count of clicks per user on a specific card*
- *Percent of unique users clicking a specific card*

For the A/B testing experimentation of the thesis the cards on the landing page are considered as the testable area

Martin (2015) suggests starting the A/B testing with simple and small tests on high-traffic pages such as the landing page. In this case studying the effects of card location on the landing page follows this suggestion. It was requested that the test is run by moving a card which directs the users to their contracts page. This card is now referred to as *contracts instructions card*. The card is currently located in the bottom row of the grid. The current version of the card grid on the landing page is presented in Figure 4.

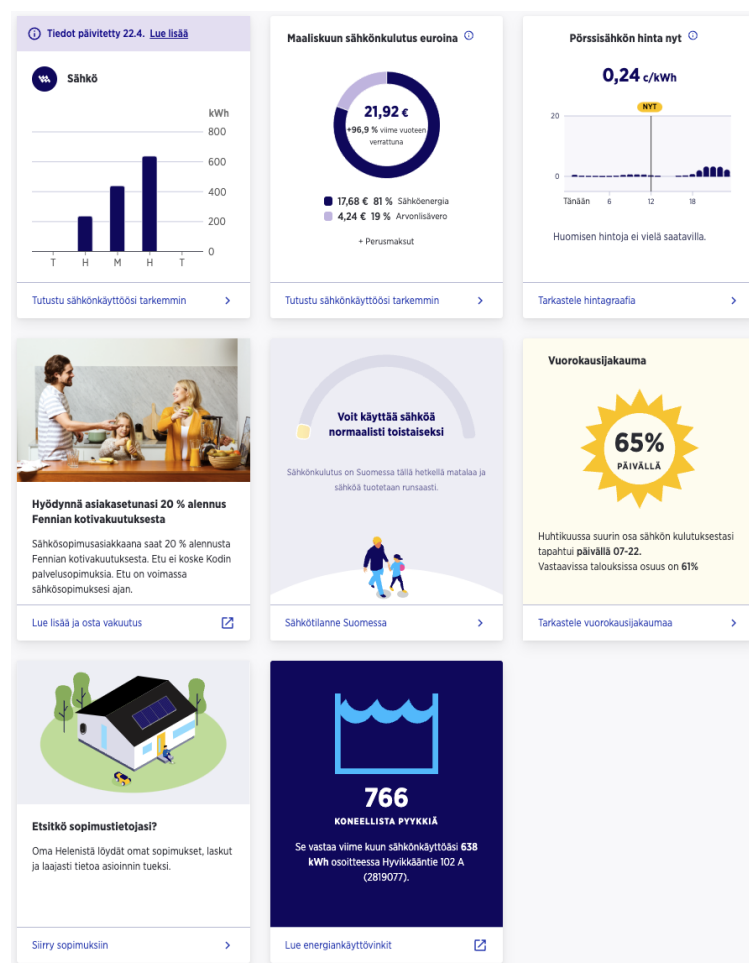


Figure 4: Current version of the card grid on the landing page. Second last item on the grid is the *contracts instructions card*.

7.2 Tool deployment

As the legal team has approved the use of the tool and the modifications to the cookies have been made the tool can be deployed. The deployment is done by following Split's step-by-step list on how to take the tool into use (Split, n.d.-f). The metric and testable area definition allows implementing event tracking on the correct places of the service. Furthermore, as Split uses specific terminology in their documentation, some of the most important terms are established in this section.

7.2.1 Important concepts

Split uses specific terms for different concepts related to their tool and A/B testing. Following concepts are helpful to understand when deploying and using the tool.

- *Feature flag*: It allows wrapping a part of the code inside the flag and enables controlling whether that code is run based on the flag state without new code deployment (Lynch, 2022a). Furthermore, the controlling can be done in a targeted way as specific as individual user level.
- *Treatments*: The test variants are called treatments. The treatments and their amount can be modified at any time (Split, 2018b).
- *Impression*: It is a record of the completed targeting decision which is created automatically every time a feature flag is evaluated (Lynch, 2022a). It contains information of the user for which the evaluation was completed, the targeting decision, targeting rule that the decision was based on, and a time stamp.

7.2.2 Choosing the SDK

The first step on the step-by-step list to use the tool is installing a Split's SDK, which functions as a decision engine and automatically monitors what variants are served to users (Stuart, 2019a). However, before installing the SDK, a suitable SDK needs to be chosen from the large list of SDKs Split offers (Stuart, 2019a). As the focus of the A/B testing experimentation is on the front-end side, the interest is in client-side SDKs. Especially the JavaScript SDK and Browser SDK seem most promising as they use JavaScript/TypeScript as their language.

The JavaScript SDK is already in use in the development process to control feature flags. The Browser SDK is built on top of the JavaScript SDK; however, Split claims it to be more optimised for web browsers (Split, 2021a). Both SDKs provide the same A/B testing functionalities in slightly varying syntax. In addition to these two SDKs, there is a third option of using Browser Suite rather than just the Browser SDK. The Browser Suite is built on top of the Browser SDK and a Browser RUM Agent (Zelaya, 2023). The RUM stands for Real User Monitoring and the agent monitors errors, page load time, and time taken for a document to be ready (Split, 2019a). Furthermore, it allows tracking to Web Vitals by Google which enables automatic collection of more events that monitor the user experience. All these different options provide

the capability of conducting A/B testing. However, as the Browser Suite provides automatic event tracking, it is chosen as the SDK for the implementation.

7.2.3 Installing and initialising the SDK

Split allows installing the SDK with either Node package manager (NPM), Yarn or CDN bundle (Zelaya, 2023). NPM is already used in the project as the package manager and therefore it is used for the installation. As the JavaScript SDK was already in use in the project and it mostly follows the same syntax, a large amount of the code is reusable and only minor changes are needed for taking the SDK into use. Listing 1 presents the code for SDK initialization when using the JavaScript SDK and Listing 2 code for initialising the Browser Suite. Only difference between the codes is located on the first row, rather than using `SplitFactory` instance the Browser Suite uses `SplitSuite` instance.

```
1 export const splitFactory: SplitIO.IBrowserSDK = SplitFactory({
2   core: {
3     authorizationKey: 'SDK_key',
4     key: 'key', // identifies users
5   },
6   startup: {
7     readyTimeout: 1.5,
8   },
9   userConsent: 'UNKNOWN'
10 });
```

Listing 1: SDK initialization with JavaScript SDK (Split, 2018d).

```
1 export const splitSuite: SplitIO.IBrowserSDK = SplitSuite({
2   core: {
3     authorizationKey: 'SDK_key',
4     key: 'key', // identifies users
5   },
6   startup: {
7     readyTimeout: 1.5,
8   },
9   userConsent: 'UNKNOWN'
10 });
```

Listing 2: SDK initialization with Browser Suite (Zelaya, 2023).

After the new SDK Browser Suite was initialised, a problem occurred when using the tool in localhost mode which should allow using the tool without contacting the Split cloud. Split claims that the Browser Suite supports the localhost mode (Split, 2021a). However, when initialising the SDK in localhost mode, it fails with an error related to the RUM Agent and Web Vitals. This breaks the development process pipeline as the UI tests use Split in localhost mode. Split's documentation does not contain advice on the error and due to time limits with the project, the use of the Browser Suite is discarded as A/B testing is also possible with the SDK already in use. The currently used SDK, JavaScript SDK, does not produce an error in the localhost mode. When the A/B testing capability is developed and validated, the move to the

new SDK can be considered again as the syntax is approximately the same in both SDKs. Thus, the possible migration should be quite effortless.

7.2.4 Creating and using feature flags

The second step on the list is to create feature flags ([Split, n.d.-f](#)). In Split, they are created on the Split's application. Initially, the feature flag is given a name and a traffic type which cannot be changed later ([Stuart, 2019b](#)). Furthermore, the owner of the flag is set and optionally tags, and a description can be specified. Once the flag is created, additional targeting rules can be set, and exposure can be controlled ([Stuart, 2019b](#)). A/B tests are run by setting the exposure of the treatments to fifty-fifty percents.

As feature flags have already been used in the project, there are multiple ready-made flags in use. However, for assessing the effects of card location, a new feature flag must be created. [Figure 5](#) shows how the feature flag is created in the Split application. The name of the feature flag is set to *SwitchCardPlacement* and the description contains information on what the flag is used for. The traffic type is set to *user*, meaning that the keys associated with the feature flag impressions correspond to user identifiers ([Split, 2018f](#)).

After creating the flag, the environment(s) needs to be initiated and the possible additional treatment can be set. In this case there is only need for the default *on* and *off* treatments. This means that when the flag is set to *on* the card placements is switched and when it is set to *off* the placements remain at the original positions as presented in [Figure 4](#). Furthermore, the treatment distribution can be set so that fifty percent of the traffic receives the *on* treatment and remaining fifty percent the *off* treatment.

The feature flag needs to be taken into use in the code by evaluating it. [Listing 3](#) contains the code for evaluating the created feature flag value. With the result of the evaluation (`true` or `false`), the elements visible on the UI can be controlled. Browser Suite uses the same syntax for feature flag evaluation ([Split, 2021a](#)). Thus, migrating to the Browser Suite does not require code changes to feature flag evaluations.

```
1 client.on(client.Event.SDK_READY, function() {
2   const treatment: SplitIO.Treatment = client.getTreatment('
3     SwitchCardPlacement');
  });
```

Listing 3: Evaluating feature flag value ([Split, 2018d](#)).

7.2.5 Sending event data

Next step is to send event data, which is then used in the metric calculations ([Split, n.d.-f](#)). Split offers three ways to send and track event data. Firstly, using the `track` method provided by the SDK client. Another way to send event data to Split is by posting it as a JSON body to `events` API provided by Split. Finally, by integrating the Split with data analytics tools such as Google Analytics or Amazon S3, events can be sent to Split ([Stuart, 2019c](#)). In this case, as the SDK client is already initialised in the code and used for feature flagging purposes, it is natural to use it also for event tracking.

Create a feature flag

Name

Use this name in your code. Once created, the feature flag name cannot be changed.

Traffic Type

Once created, the traffic type cannot be changed.

Owners

Tags (optional)

Use tags to organize your feature flags by team, feature, or however you'd like.

Description (optional)

The feature flag is for switching the placement of a card on the landing page for feature A/B testing purposes.

External objects

There are no integrations configured to associate external objects with your organization's feature flags. [Configure](#) an integration to connect an external object to this feature flag.

Figure 5: Creating a new feature flag on Split's application ([Split, n.d.-c](#)).

In this A/B testing experimentation, the effect of card placement on the clicking rate of the card is studied. Thus, event tracking on whether a user has clicked the card must be implemented. To enable tracking, a function is created which has the same parameters as Split's `track` method. Listing 4 presents the code for the function. The `track` method is a method provided by the JavaScript SDK ([Split, 2018d](#)). However, Browser Suite also uses the same method, thus future migration does not require code changes to event tracking ([Split, 2021a](#)).

```
1 export const trackEvent = (event: SplitIO.EventData) => {
2   splitFactory.client().track(event.trafficTypeName || 'user',
3     event.eventTypeId, event.value, event.properties);
}
```

Listing 4: Function for event tracking.

Reason for creating the event tracking function is to minimise the code and make possible migrations easier. As now only the function needs to be exported and implemented in the parts where there is a desire to track an event. Thus, if the SDK is changed, the only place which needs modifications is the original file which creates the tracking function. An example code implementation for tracking the *contracts instructions card* is presented in Listing 5. The naming of the event follows the recommended best practices of Split ([Split, 2019c](#)).


```

1 import { trackEvent } from '@lib/utils/feature-flags';
2
3 <Link onClick={() => trackEvent({eventId: "card.
  contractsInstruction.clicked"})} />

```

Listing 5: Tracking the click events of *contracts instructions card*.

7.3 Metric creation

The metrics in Split are created on their application (Ball, 2023). Each metric has a name and its owners (Ball, 2023). Additionally, tags or descriptions can be set for a metric. Furthermore, whether the value of the metric is desired to increase or decrease and what traffic type it uses is chosen (Ball, 2023). Finally, how the metric is measured and what event it uses to calculate the metric are determined. The different ways to measure the metric are listed on Section 6. For instance, when creating a metric for tracking the clicking activity for the *contracts instructions card*, the name can be set to *FP card clicks: ContractsInstruction*. The following description can be added to clarify the metric: *Tracks the count of clicks of the contracts instructions card (located on landing page) per user*. Finally, increase is chosen as desired impact and the metric is set to measure the count of events per user where the event is defined as `card.contractsInstruction.clicked`. For each card, the metric for tracking its clicking activity can be created using this pattern. In addition, for each card metric that tracks the percent of unique users clicking the card is created.

7.4 Validation

Now that the tool is deployed and event tracking and metric calculation are implemented, the tool can be validated using an A/A test. Split offers a guide for conducting A/A testing on their documentation (Split, 2019d). For the test, a new feature flag needs to be created with a targeting rule that divides the traffic equally to two treatments. However, in this case even though there are two treatments, their effect will be the same for the user as the A/A test tries to verify that the traffic is randomly distributed, and that the setup works properly (Kohavi & Longbotham, 2023).

After creating the feature flag, it needs to be taken into use in the code (Split, 2019d). Split recommends running the A/A test for at least a full week (Split, 2019d). This follows the same recommendation as made by Martin (2015). Furthermore, as Split offers automatic sample ratio mismatch detection, it helps to further validate that the tool is working correctly (Split, 2018e).

By following the guide by Split, a new feature flag, *AAtesting*, is created with the default targeting rule of fifty-fifty split. The feature flag is taken into use in the code with zero effects on the UI; both users with the treatment *on* and *off* will have the same UI layout.

After running the A/A test in the production, the event data is sent to Split. Figure 6 displays the tracked events on the Split application after four hours have passed in the production environment. Split correctly divides the users to two parts and exposes

them to the two different treatments. Figure 7 displays that after four hours, there are around seven hundred unique users exposed to the treatments and the split is almost at 50%. However, the metrics impact tab shows that no events have come in, for reference see Figure 8. The cards direct to the metrics impact guide (Split, 2018g) which contains a troubleshooting guide for the N/A status. However, none of the situations seemed to apply. Thus, a contact to the support was made, which provided an answer within a working day. It revealed that the problem is with the key field in the event data structure. Due to the problem, the event data cannot be associated with the impressions and therefore the results cannot be calculated.

Event type ID	Traffic types	Count (Last 7 days)	Last received at	Action
card.consumptionSummar...	user	204	2024-05-23 04:06:23 PM	View live tail View metrics
card.spotPrices.clicked	user	32	2024-05-23 04:05:29 PM	View live tail View metrics
card.contractsInstruction.c...	user	31	2024-05-23 03:34:29 PM	View live tail View metrics
card.expensesDonutChart....	user	31	2024-05-23 04:00:45 PM	View live tail View metrics
card.renewalNotification.cl...	user	19	2024-05-23 04:06:27 PM	View live tail View metrics
card.insight.tracked	user	4	2024-05-23 03:02:35 PM	View live tail View metrics
card.electricityStatusIndic...	user	3	2024-05-23 03:02:18 PM	View live tail View metrics
card.timeOfDayInsight.clic...	user	2	2024-05-23 02:49:18 PM	View live tail View metrics
card.fenniaAd.clicked	user	1	2024-05-23 12:23:40 PM	View live tail View metrics

< 1 - 20 >

Results per page: 20 ▾

Figure 6: Registered events after A/A test has been run in production for 4 hours (Split, n.d.-c).

Sample ratio: Valid

Your feature flag has a valid sample ratio based on the treatments and targeting rule you selected.

Targeted ratio:

50.00% in “on”

50.00% in “off”

Current ratio:

49.06% (341 users)

50.94% (354 users)

Go read more about [sample ratio](#) .

Figure 7: A/A test sample ratio after four hours (Split, n.d.-c).

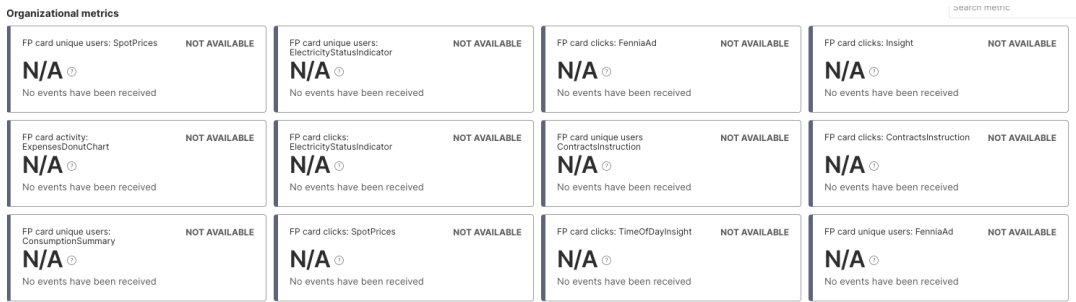


Figure 8: Metrics impact tab in A/A test showing N/A status (Split, n.d.-c).

Next step is to fix the problem, so that the Split’s track method is associated with the correct key. After fixing it, the metric calculations are already visible in the testing environment, for reference see Figure 9. This suggests that the tool is working and can be used in A/B testing. Furthermore, Split claims that running multiple tests simultaneously is possible (Split, 2020b) (Nassi & Jewkes, 2021). This is assessed by running the A/A test and the A/B test concurrently. In this case, the results should not be identical and the A/A test results should show inconclusive results.

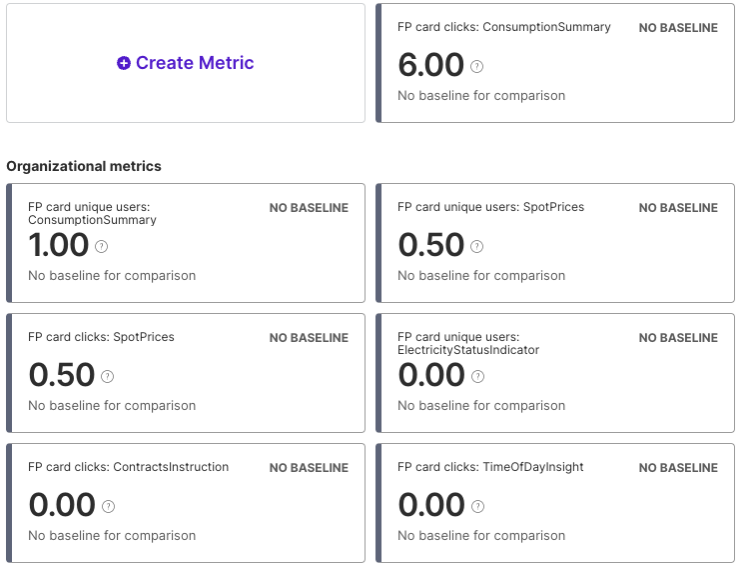


Figure 9: Metrics impact tab in testing environment after fixing the problem found in A/A testing (Split, n.d.-c).

The results of the second A/A test generated by Split for PDF are presented in Appendix A. Two of the cards did not have enough activity to calculate the metric results. However, all the calculated results are inconclusive, excluding *spot prices card* click activity per user. As all other results were inconclusive, the result for *spot prices card* is most likely a false positive. The test uses a confidence level of ninety-five percent, therefore there is five percent probability that a metric result is a false positive. To confirm this, more A/A tests should be run. However, due to the time restrictions

of this project, there is no possibility to run for instance ten more A/A tests. The A/A test was run for seven days which is the minimum recommended running time. More reliable results can be acquired by running the test for two weeks.

Overall, as all but one of the metric results were inconclusive, the test setup randomly divides the users into two groups and calculates the metric results. Thus, based on the A/A tests the setup is correctly working. Furthermore, as the A/A test was run concurrently with the A/B test affecting *contracts instruction card*, and the results of A/A test did not show any impact on the activity of that card, running multiple tests simultaneously was possible.

7.5 Constructing a hypothesis

Next step for conducting an A/B test is to construct a hypothesis. Both Split and literature support this step (Split, 2021b) (Martin, 2015) (Siroker et al., 2013). Split offers both simple and advanced templates for creating a hypothesis (Split, 2021b). As the A/B testing experiment is concerned with a relatively simple problem, the simple template is suitable for this case. Thus, by using the Split's simple template (Split, 2021b), a following hypothesis can be constructed:

1. Because we observed that the cards located in the bottom row are far away from the user's initial eyesight and/or may be hidden.
2. We expect that by moving a card to the top row from the bottom row will cause users to click that card more.

Constructing this hypothesis allows better comprehension of what the test aims to determine (Siroker et al., 2013).

7.6 Creating the treatment variant

Now that a hypothesis is constructed, the treatment variant for the A/B test can be created. The hypothesis expects that moving a card from the bottom row increases the clicking activity of the card. Thus, the treatment variant should differ from the control variant only by moving one of the cards, in this case the *contracts instructions card*, from the bottom row to the top row. Figure 10 presents the landing page card grid of the treatment variant.

7.7 Running the A/B test

Running the A/B test contains the same steps as in running the A/A test. However, in this case the feature flag controls the position of the *contracts instructions card*. The targeting rule is set to a fifty-fifty split and the test is allowed to run for seven days. During the running, the metric impact tab is followed to confirm that the test is working, and metrics are being calculated.

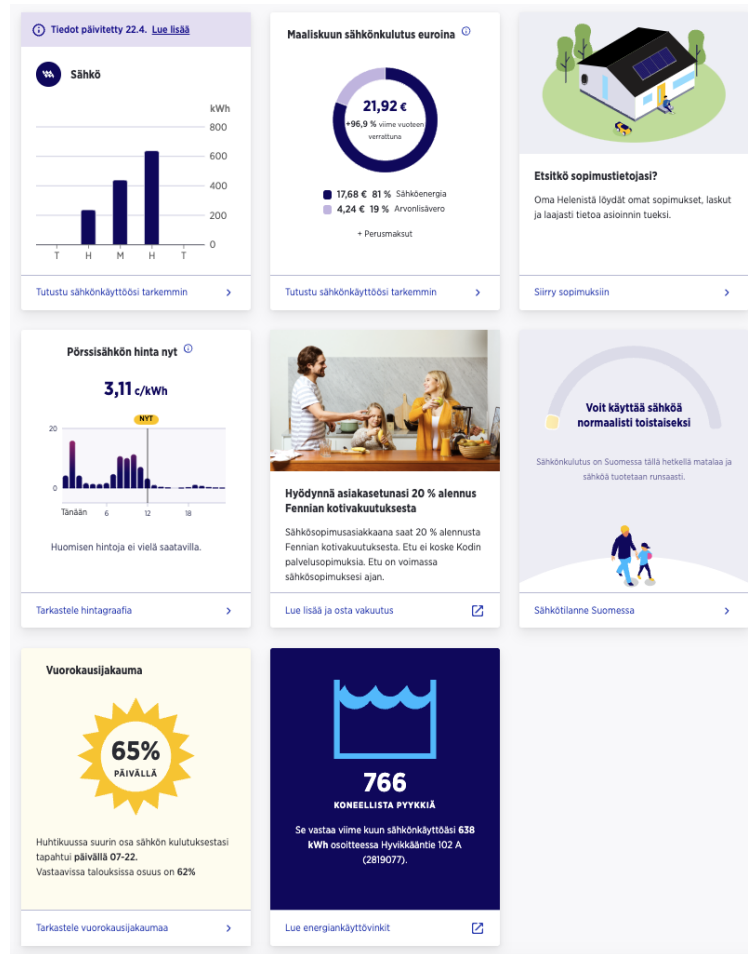


Figure 10: Card grid of the landing page for the treatment variant. The *contracts instructions card* has been moved to the top row of the grid.

7.8 Analysing results

The overview of all A/B test results generated by Split for PDF export is presented in Appendix B. More detailed results for the first three cards on the original grid and the *contracts instructions card* are presented in Table 6. The results in the table include the metric values and the p-values as well as possible impacts on the metrics the new treatment had.

The results indicate that the position of the card affects the activity significantly. The impact on clicking activity of the moved card increased fifty-five percent when examining the number of unique users clicking the card and forty-nine percent when assessing the number of clicks a unique user makes. Concurrently, the impacts of the second card on the grid, which in this experiment did not move, decreased. For unique users clicking the card the impact decreased twenty-one percent and for users clicking the card the impact decreased eighteen percent. Two of the cards did not receive enough activity to calculate results. As the interest of the test was focused on the moved card, it was acceptable that the results for those cards could not be

calculated. However, if there is an interest in the results of those cards, the test should be kept running until results can be calculated. In this case, the test was run for seven days, which was the minimum suggested running time (Martin, 2015). More reliable results can be acquired by allowing the test run for two whole weeks.

Overall, the A/B testing experiment was successful, and it gives a good starting point to continue experimenting. It indicated that the position of the card can have a large effect on the clicking activities. However, more tests on the positions of the cards should be done to confirm the quantifiable effects of card position. Furthermore, based on the second A/A test and A/B test results, running concurrent experiments is possible. This enables the capability to run tests for each new feature even if they are released simultaneously.

Table 6: A/B test results for four landing page cards.

Metric	Result	Treatment metric value	Control metric value	p-value
Percent of unique users clicking contracts instructions card	+54.78% impact	0.07	0.04	0.0
Number of clicks on contracts instructions card per user	+48.89% impact	0.07	0.05	0.0
Percent of unique users clicking consumption summary card	Inconclusive	0.24	0.24	0.77
Number of clicks on consumption summary card per user	Inconclusive	0.55	0.63	0.05
Percent of unique users clicking expenses donut card	-21.10% impact	0.07	0.08	0.04
Number of clicks on expenses donut card per user	-18.21% impact	0.08	0.1	0.04
Percent of unique users clicking spot prices card	Inconclusive	0.03	0.04	0.05
Number of clicks on spot prices card per user	Inconclusive	0.07	0.09	0.44

7.8.1 Observations

During the tool deployment phase there was a problem with taking a new SDK into use. The problem only occurred in localhost mode, and more specifically when initiating

the SDK in the localhost mode. Same problem did not occur with the old SDK, which suggests that there is a problem with the Browser Suite. However, the documentation did claim that localhost mode is usable with the Browser Suite (Zelaya, 2023) and there was no mention of the problem in the troubleshooting guides (Split, 2019f). However, as the old SDK also allows conducting A/B testing, it could be used for the experiment. However, in the future taking the Browser Suite into use would be beneficial as it automatically tracks the Web Vitals, thus allowing to gather more information about the service. Fortunately, the method syntax between the SDKs is the same which enables easy migration (Split, 2018d) (Zelaya, 2023).

As Split was already used for feature flags, there already existed a practice for taking them into use in the code. Thus, the effort needed to use feature flags was quite minimal. There was no event tracking previously and therefore it had to be implemented. However, implementing event tracking required only a couple rows. Now the developers can track more events by just writing a few lines of code per event. Overall, the event tracking with Split is surprisingly easy, requires little effort, and can be easily verified on the application. Furthermore, creating the metrics is quite effortless on the application as it automatically detects the available events.

Beginning to run an A/A or an A/B test is simple especially if no new metrics need to be created. Running a test only requires creating a flag, which is already a common practice when implementing new features and setting the targeting rule to fifty-fifty split. If no suitable metrics are implemented, the developer needs to create a metric on the Split application and possibly deploy event tracking to the code. However, both processes are quite simple and fast.

The initial validation revealed a problem with the setup, which prevented associating the events with the impressions. Due to this, initially no metric results were calculated. The problem was solved by contacting customer support, which provided answers and instruction within a working day. After fixing the problem, the setup seems to work correctly. The results are easy to read and understand. However, there is no built-in method for ending the experiment after a certain time. Thus, to export the results after the seven days exactly, the tester needs to remember to go to the Split application and recalculate and export the results at that time.

In conclusion, there were problems during the experimentation phase, but the benefits and ease of use outweigh these. For instance, as the development process already uses the feature flags to implement new features, it is possible to incorporate the A/B testing with the process extremely effortlessly. However, as only a simple experiment has been conducted, possible additional problems may occur when taking the A/B testing process into teamwide and long-term use.

8 Discussion

The objective of this thesis was to develop A/B testing capability using a third-party A/B testing tool for the development process used to develop a digital service for users of an electricity sales company. The A/B testing capability was evaluated with simple A/A and A/B experiments with the digital service. To support the process of developing the A/B testing capability, five research questions were identified and they were answered throughout the thesis. In this section the results of each question are discussed as well as whether the objective of the thesis has been met. Furthermore, future of A/B testing and online experimentation is discussed.

8.1 Research questions

The results of each research question are summarised and discussed and possible problems identified.

8.1.1 What is the development process in use and how A/B testing with a third-party tool can be incorporated with that?

The development process uses agile and DevOps practices. In addition, feature flags are used for implementing and controlling features in digital service development. Due to their ability of controlling the UI, they can be easily utilised in A/B testing. Thus, conducting A/B testing with feature flags seemed the most suitable way to incorporate online experimentation with this development process.

There are specific technologies and programming languages in use in the project which restrict the tool selection; it must support JavaScript/TypeScript and should support Python and Flutter. In addition, the tool must abide by the GDPR regulations, and it would be beneficial if event tracking can easily be bound to user consent.

More detailed analysis of the development process and description of the digital service used in the experimentation is in Section 3. In conclusion, the characteristics of the development process and technologies in use as well as the geographic location of the organisation need to be studied when incorporating any new tool or a practice as they can apply restrictions. Furthermore, in this case utilising feature flags for A/B testing minimised the number of new steps added to the process. However, this solution was process specific and for a different development process this approach may not be suitable.

8.1.2 What aspects should be considered when choosing a tool for the development process?

When choosing a new tool for a development process there are common criteria which should be considered. [Jadhav and Sonar \(2009\)](#) provide a comprehensive list in their paper of the general criteria. However, in addition to the common criteria, the focus area of the tool and the used development process should be noted as they can provide additional criteria.

Sections 5.2 and 5.4 discuss in detail the restrictions and assessment criteria relevant for choosing an A/B testing tool for this development process. In conclusion, the most restricting criteria is the present regulations (GDPR) and the required programming language or technology support. Furthermore, offered functionalities and help documentation should be considered. Finally, as the focus area of the tool is A/B testing it caused specific criteria such as how the events are tracked and how easy running an A/B test is with the tool.

Price of a tool was not included in the assessment criteria. However, the price of a tool can be an important and restricting criterion. In this case, they were not included because the tools do not disclose their pricing on their sites but require a price request to be sent to them. If more studies on A/B testing tool comparison is conducted, the price should be included. Furthermore, if the price is an important criterion for the organisation selecting a tool, it needs to be included in the comparison.

Finally, as the tools are offered by third-party organisations, the availability of the tool can change. For example, the organisation may discontinue the tool. Or other changes within the organisation or the tool can occur. For instance, while working on this thesis, the organisation behind the chosen tool, Split, was sold to another company (Bell, 2024). Initially, this does not seem to affect the use of the tool. However, the possible changes to tool availability should be noted when choosing a tool.

8.1.3 How tools for A/B testing differ from each other and what tool is the most suitable for the process?

The comparison of different A/B testing tools was done in two parts. The initial comparison between six different tools (AB Tasty, Adobe Target, Firebase, Optimizely, Split and VWO) is conducted in Section 5.3. It focused on the offered functionalities and supported programming languages. The results display that most of the tools offer the same or similar basic functionalities. These include for instance visual editor, multivariate testing, targeting and automation. However, the comparison only contained six tools, thus the comparison should be expanded to confirm this. Nevertheless, this gives an indication that the tools do not differ greatly when considering the functionalities.

The supported programming languages of the different tools in the initial comparison had more variation. For instance, only AB Tasty, Optimizely and Split supported all three languages (JavaScript/TypeScript, Python and Flutter) while offering feature flag and A/B testing capability. The variability of supported programming languages indicates that when choosing a tool for an existing product with specific languages, the language support should be verified first. Furthermore, if there is a possibility that additional programming languages need to be supported it should be considered already when choosing a tool.

Based on the initial comparison AB tasty, Optimizely and Split were chosen for the more detailed comparison conducted in Section 6.4. The comparison further validated that the tools and their functionalities are similar. For instance, Split and Optimizely seem to offer almost identical functionality. However, there were aspects such as data exportation and event tracking where Split is slightly superior to Optimizely.

Furthermore, Split was already used in the project for feature flags, thus the deployment and use should be easy as the team members are familiar with the tool. Due to these reasons, Split was considered as the most suitable A/B testing tool for the process.

The initial comparison included six different tools and three tools were compared in more detail. Thus, the sampling was quite narrow, and no comprehensive conclusions can be made. To confirm that the A/B testing tools in general offer the same functionalities, the number of tools in the comparison need to be increased. However, these initial results indicate that and can be considered as a starting point for a more exhaustive comparison study.

8.1.4 What steps are needed for implementing A/B testing in practice?

In Section 2.7 the steps of implementing an A/B test based on literature are presented. These steps are used as a base and combining them with instructions offered by Split the steps needed to implement A/B testing in practice were identified in Section 4.2.

Validation is a crucial step when developing A/B testing capability, as it allows testing whether the A/B testing setup is working correctly. When using an A/A test as the validation method, the validation can be conducted without affecting the user experience. In this paper, the validation step revealed a small but major problem in the setup which prevented computing the metric calculations.

Based on the identified steps two guides are created for the development team containing A/B testing steps. First guide (see Appendix C) presents the general A/B testing steps and it gives the team members an idea of what A/B testing process should contain. As this is a general guide, it can be utilised in different processes and projects. Second guide (see Appendix D) offers detailed steps on how to run A/B tests with Split and is targeted for the team members who are directly responsible for running the tests on Split.

The first guide contains 7 general steps (see Appendix C). In practice, the process may be too heavy especially when deploying rapidly new features. Studying how organisations and teams actually conduct A/B testing could reveal whether the generally advised steps work in practice or whether teams have established their own process.

8.1.5 Is the chosen tool suitable to use for the A/B testing in the development process?

During the experimentation phase problems and observations on the ease of use of the tool were noted to analyse whether the tool was suitable. These observations are listed in Section 7.8.1. Overall, the tool is simple and easy to use, and the benefits outweigh the problems. Furthermore, as the tool was already in use for feature flagging the additional effort needed from the team members to use the tool for A/B testing is quite minimal especially if the suitable metrics already exist.

However, the actual suitability and how easily the A/B testing can be incorporated with the development process can be properly analysed over a longer time period where the team members start conducting A/B tests themselves. Thus, the long-term results for this question remain unknown.

8.2 Development of A/B testing capability

The objective in this thesis was to enable A/B testing capability for a development process and more specifically for a project developing a digital service for the customers of an electricity sales company. The research questions supported the thesis process and helped to reach the objective. The A/B testing capability was evaluated by conducting an A/B testing experiment and its results were analysed in Section 7. Overall, the capability of conducting A/B testing in the project is achieved. The tool, Split, was deployed and an experimental test has been run which showed that the tool is working. Furthermore, in the experiment two simultaneous tests, an A/A test, and an A/B test, were run successfully. This confirms that the tool has the capability of running concurrent tests. This accelerates the testing process, as there is no need to wait for another test to finish when running a new test.

The capability is still in preliminary form and by continuing to conduct A/B testing and adopting the A/B testing mentality in the development process, more extensive capability can be achieved. Furthermore, simple metrics were used in the experiment, as its purpose was only to confirm that A/B testing is possible with Split. Thus, the next step would be to identify and implement OEC metric(s) that measure the long-term business goals and user satisfaction.

Furthermore, the capability was developed for the web and more specifically for the front-end. Thus, there is no capability to conduct A/B testing on different platforms when developing the digital service. In future, the possibility to move to using Split for A/B testing on back-end and mobile should be considered. Furthermore, the deployment of Browser Suite should be reconsidered now that there is verification that the tool can be used to conduct A/B testing. By deploying the Browser Suite, additional events such as Web Vitals can be tracked automatically which provides more data for A/B tests (Zelaya, 2023).

Overall, the capability is still in initial form. Next step is to educate the team members on the A/B testing and start conducting A/B testing as part of the development process. By fully adopting the A/B testing and experimentation mentality within the team the benefits can be reached (Koning et al., 2022). To help the team members understand what A/B testing means, a concise summary is created. The summary is presented in Appendix E. In addition, to create more concrete examples and motivate the team members a demo showing how to conduct A/B testing and how to read the results should be presented. Studying and tracking how well the team has adopted the A/B testing mentality and whether it is effectively used should be done to verify that A/B testing capability has been truly developed in a long-term setting.

8.3 Future of A/B testing and online experimentation

The availability of different A/B testing tools has increased, the costs of experimentation decreased and the competition between online services is increasing (Koning et al., 2022) (Martin, 2015). Thus, there is reason to believe that the online experimentation becomes even more common within different sized organisations. However, due to already rising discussion on ethical problems (Jiang et al., 2019) and data regulations

such as the GDPR ([European Parliament & Council of the European Union, 2016](#)), more studies on how to conduct online controlled experimentation ethically while protecting the user data needs to be done.

Furthermore, currently there is a large technological turning point of using generative artificial intelligence (AI) and with high confidence, it also affects the world of online experiments. For instance, AI could be used to generate variant ideas or summarise the results of tests. [Martin \(2015\)](#) believes that machine learning (ML) in general can be utilised in A/B testing by generating the test ideas and speeding up the testing process. Its use is already visible, as Optimizely uses ML to accelerate test results ([Optimizely, 2024d](#)). Additionally, generative AI can enhance the visual editors such as WYSIWYGs by generating different variants and their code. This would allow team members with no programming or design skills to conduct A/B testing.

9 Conclusion

In this thesis, A/B testing capability using a third-party tool was developed using the research questions as a guide for the development process. Figure 11 visualises these steps. The process steps are not case specific and therefore the process can be used as a guide for other projects or organisations for developing A/B testing capability.

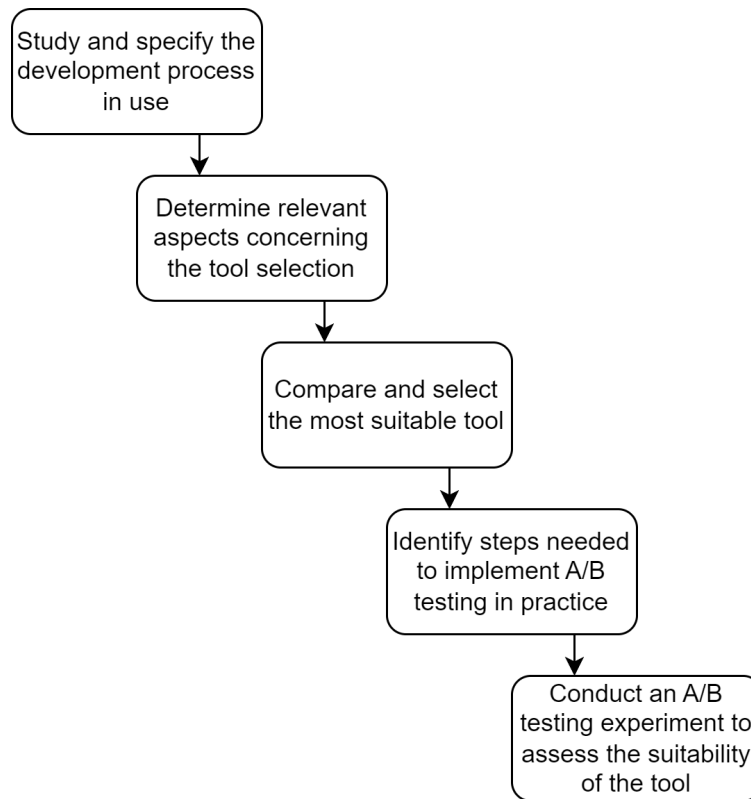


Figure 11: Steps for developing A/B testing capability.

The first step in the process is to study and specify the development process in use and the product for which the A/B testing capability is developed. This helps in determining restrictions for the tool, such as programming language or regulation support, and how the tool and A/B testing process can be incorporated with the development process. In this case, the process used feature flags as a development tool and different A/B testing tools utilise them for separating the different variants. Thus, the approach where the selected tool offers both feature flag and A/B testing functionalities was used.

The next step determines aspects that need to be considered when selecting a tool for a development process. Using the literature various general aspects that apply to selecting any software tool could be determined. In addition to those, case specific criteria need to be included. For instance, the tool must comply with the GDPR as the organisation is Finnish. Furthermore, as A/B testing is the focus area of the tool, it provides additional criteria such as how easy it is to run an A/B test with the tool and what type of metric calculation it supports.

The third step concerns with the actual comparison and selection of the A/B testing tools. In this thesis, a systematic comparison approach based on literature was used. The comparison included initial comparison which was used to eliminate some of the tools and more in-depth comparison to select the most suitable tool. Based on the comparisons, most of the A/B testing tools seem to offer the same basic functionalities, while the supported programming languages differ greatly. However, the sampling was small and broader comparison needs to be completed to make concrete conclusions. Based on the assessment criteria used in the in-depth comparison, Split was the most suitable tool for the development process. Furthermore, it was already used in the process which allowed easy use and deployment.

In the fourth step the steps needed for implementing A/B testing in practice are identified. Literature was used to determine the general steps of A/B testing. However, as the goal of the thesis was to develop A/B testing capability with a third-party tool, steps for deploying and validating the tool were included. Furthermore, initial preparations had to be made before starting to develop the capability. For instance, as A/B testing uses user data, approval that the tool and A/B testing comply with the GDPR was requested. When developing A/B testing capability for a project or an organisation, time should be reserved for this. The A/B testing setup was validated using A/A testing, which allows validating the tool without affecting the user experience. In this case, the A/A test revealed a problem with the setup which prevented result calculation. The problem was fixed, and the setup was further validated with a second A/A test.

The fifth step aims to determine whether the chosen tool is suitable for A/B testing in the development process. The suitability was evaluated by conducting a simple A/B testing experiment with the digital service following the steps identified in the previous step. The experiment was successful and confirmed that the A/B testing capability is developed. Furthermore, observations on the ease of use and potential problems with the tool were noted throughout the experimentation. Overall, the tool has a few minor problems, however the positives, such as ease of use and easy integration with the process, outweigh these. By adopting the A/B testing as a part of the development process, the long-term results for tool suitability can be obtained.

In conclusion, using these steps the A/B testing capability was developed for a software process used to develop a digital service for an electricity sales company. Now the developers of the team can with low effort run A/B tests when releasing new features. The capability also allows designers to create and evaluate design variations. Simple metrics have already been created to measure user behaviour in the A/B tests. However, more metrics and metrics that measure long-term organisational goals need to be implemented to expand the A/B testing capability. Furthermore, the next step is to adopt the A/B testing mentality and incorporate A/B tests as a part of the development process. To achieve this, the team members need to be educated on what A/B testing is and how to run A/B tests using Split. By constantly experimenting with A/B tests and truly adopting the A/B testing mentality, true A/B testing capability and its full benefits such as performance increase can be achieved.

References

- AB Tasty. (n.d.-a). *Combine specificity with scale using Content Personalization*. Retrieved May 15, 2024, from www.abtasty.com/content-personalization/
- AB Tasty. (n.d.-b). *Data-driven experiences with feature experimentation*. Retrieved May 15, 2024, from www.abtasty.com/feature-experimentation/
- AB Tasty. (n.d.-c). *Faster and safer releases with AB Tasty Rollouts*. Retrieved May 15, 2024, from www.abtasty.com/rollouts/
- AB Tasty. (n.d.-d). *Feature Experimentation Rollouts - Developer Docs*. Retrieved May 15, 2024, from <https://docs.developers.flagship.io/>
- AB Tasty. (n.d.-e). *Feature Experimentation Rollouts - FAQ*. Retrieved May 15, 2024, from flagship.zendesk.com/hc/en-us/sections/360003019199-FAQ
- AB Tasty. (n.d.-f). *Feature Experimentation Rollouts - Help Center home page*. Retrieved May 15, 2024, from <https://flagship.zendesk.com/hc/en-us>
- AB Tasty. (n.d.-g). *Feature Experimentation Rollouts - VIDEO TUTORIALS*. Retrieved May 15, 2024, from flagship.zendesk.com/hc/en-us/sections/360005486380-VIDEO-TUTORIALS
- AB Tasty. (n.d.-h). *Guide buyers to their desires and push priorities*. Retrieved May 15, 2024, from www.abtasty.com/intelligent-search/
- AB Tasty. (n.d.-i). *Turn frustrations into growth foundations with Web Experimentation*. Retrieved May 15, 2024, from www.abtasty.com/web-experimentation/
- AB Tasty. (n.d.-j). *We are AB Tasty*. Retrieved May 15, 2024, from www.abtasty.com/about-us/
- AB Tasty. (n.d.-k). *What's our recommendation? Highlight and delight*. Retrieved May 15, 2024, from www.abtasty.com/recommendations/
- AB Tasty. (2020). *Feature Experimentation Rollouts - Reporting - A/B Test*. Retrieved May 15, 2024, from <https://flagship.zendesk.com/hc/en-us/articles/360014654940-Reporting-A-B-Test>
- AB Tasty. (2021a). *Feature Experimentation Rollouts - Configuring an A/B Test*. Retrieved May 15, 2024, from <https://flagship.zendesk.com/hc/en-us/articles/360022108620--Configuring-an-A-B-Test>
- AB Tasty. (2021b). *Feature Experimentation Rollouts - Configuring KPIs*. Retrieved May 15, 2024, from <https://flagship.zendesk.com/hc/en-us/articles/360021033379--Configuring-KPIs>
- AB Tasty. (2021c). *Feature Experimentation Rollouts - Exporting reporting data*. Retrieved May 15, 2024, from flagship.zendesk.com/hc/en-us/articles/360021861139--Exporting-reporting-data
- AB Tasty. (2021d). *Feature Experimentation Rollouts - Quick start guide*. Retrieved May 15, 2024, from <https://flagship.zendesk.com/hc/en-us/articles/360021032480-Quick-start-guide>
- AB Tasty. (2021e). *Feature Experimentation Rollouts - [Troubleshooting] How can I know my test is reliable and my data significant enough to be*

- analyzed? Retrieved May 15, 2024, from flagship.zendesk.com/hc/en-us/articles/360020939959--Troubleshooting-How-can-I-know-my-test-is-reliable-and-my-data-significant-enough-to-be-analyzed
- AB Tasty. (2021f). *Feature Experimentation Rollouts - Understanding the "Chances to win" indicator*. Retrieved May 15, 2024, from <https://flagship.zendesk.com/hc/en-us/articles/360021434400-Understanding-the-Chances-to-win-indicator>
- AB Tasty. (2022). *Feature Experimentation Rollouts - AB Tasty Demo - How to use it*. Retrieved May 15, 2024, from <https://flagship.zendesk.com/hc/en-us/articles/6535217756956-AB-Tasty-Feature-Experimentation-Rollouts-Demo-How-to-use-it>
- AB Tasty. (2024a). *JavaScript, Node, Deno V3.2.X*. Retrieved May 15, 2024, from <https://docs.developers.flagship.io/docs/javascript-node-deno-v3-2-x>
- AB Tasty. (2024b). *Universal Collect Documentation*. Retrieved May 15, 2024, from docs.developers.flagship.io/docs/universal-collect-documentation
- AB Tasty. (2024c). *Your Digital Experience Optimization Platform Partner*. Retrieved May 15, 2024, from www.abtasty.com/
- Adobe for Business. (2024). *Adobe Target: A/B Testing Optimizations*. Retrieved May 15, 2024, from business.adobe.com/ai/products/target/adobe-target.html
- Aijaz, A., Stuart, T., & Jewkes, H. (2018). *Understanding Experimentation Platforms* (1st edition ed.). O'Reilly Media, Inc.
- Ball, L. (2023). *Split Help Center - Metrics*. Retrieved May 15, 2024, from <https://help.split.io/hc/en-us/articles/22005565241101-Metrics>
- Bell, B. (2024). *SPLIT + HARNESS: UNLOCKING THE FUTURE OF SOFTWARE DELIVERY*. Retrieved June 5, 2024, from <https://www.split.io/blog/split-joins-forces-with-harness/>
- Benbunan-Fich, R. (2017, 7). The ethics of online research with unsuspecting users: From A/B testing to C/D experimentation. *Research Ethics*, 13(3-4), 200–218. doi: 10.1177/1747016116680664
- Bjarnason, E., Åberg, P., & Ali, N. b. (2023, 3). Software selection in large-scale software engineering: A model and criteria based on interactive rapid reviews. *Empirical Software Engineering*, 28(2), 51. Retrieved from <https://link.springer.com/10.1007/s10664-023-10288-w> doi: 10.1007/s10664-023-10288-w
- Datanyze. (2024). *A/b testing software market share*. Retrieved May 15, 2024, from www.datanyze.com/market-share/testing-and-optimization--56
- European Parliament, & Council of the European Union. (2002). *Directive 2002/58/ec of the european parliament and of the council of 12 july 2002 concerning the processing of personal data and the protection of privacy in the electronic communications sector (directive on privacy and electronic communications)*. Retrieved May 15, 2024, from eur-lex.europa.eu/eli/dir/2002/58/oj

- European Parliament, & Council of the European Union. (2016). *Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation)*. Retrieved May 15, 2024, from data.europa.eu/eli/reg/2016/679/oj
- Fabijan, A., Dmitriev, P., Olsson, H. H., & Bosch, J. (2017, 8). The Benefits of Controlled Experimentation at Scale. In *2017 43rd euromicro conference on software engineering and advanced applications (seaa)* (pp. 18–26). IEEE. Retrieved from <http://ieeexplore.ieee.org/document/8051322/> doi: 10.1109/SEAA.2017.47
- Firebase. (2024a). *Firestore a/b testing*. Retrieved May 15, 2024, from firebase.google.com/docs/ab-testing
- Firebase. (2024b). *Google's Mobile and Web App Development Platform*. Retrieved May 15, 2024, from firebase.google.com/
- Fortum. (n.d.). *Oma Fortum*. Retrieved May 17, 2024, from <https://www.fortum.fi/kotiasiakkaille/oma-fortum-palvelu>
- Gartner. (2024). *A/b testing tools reviews and ratings*. Retrieved May 15, 2024, from www.gartner.com/reviews/market/a-b-testing-tools
- Helen. (n.d.). *Oma Helen palvelee ympäri vuorokauden*. Retrieved May 17, 2024, from <https://www.helen.fi/sahko/sahkoasiakkaalle/oma-helen>
- Jadhav, A. S., & Sonar, R. M. (2009, 3). Evaluating and selecting software packages: A review. *Information and Software Technology, 51*(3), 555–563. Retrieved from <https://linkinghub.elsevier.com/retrieve/pii/S0950584908001262> doi: 10.1016/j.infsof.2008.09.003
- Jiang, S., Martin, J., & Wilson, C. (2019, 1). Who's the Guinea pig? Investigating online A/B/N tests in-the-wild. In *Fat* 2019 - proceedings of the 2019 conference on fairness, accountability, and transparency* (pp. 201–210). Association for Computing Machinery, Inc. doi: 10.1145/3287560.3287565
- Jurvelin, K. (2024). *Analyysi - joko pian käynnistyvät sähkömarkkinoilla hullut päivät? sähkön myyntiyhtiöitä niputetaan jälleen yhteen. Talouselämä*. Retrieved May 17, 2014, from <https://www.talouselama.fi/uutiset/joko-pian-kaynnistyvat-sahkomarkkinoilla-hullut-paivat-sahkon-myyntiyhtioita-niputetaan-jalleen-yhteen>
- Kohavi, R., & Longbotham, R. (2023). Online Controlled Experiments and A/B Tests. In *Encyclopedia of machine learning and data science* (pp. 1–13). New York, NY: Springer US. Retrieved from https://link.springer.com/10.1007/978-1-4899-7502-7_891-2 doi: 10.1007/978-1-4899-7502-7_{_}891-2
- Kohavi, R., Longbotham, R., Sommerfield, D., & Henne, R. M. (2009, 2). Controlled experiments on the web: Survey and practical guide. *Data Mining and Knowledge Discovery, 18*(1), 140–181. doi: 10.1007/s10618-008-0114-1
- Kompella, K. (2015). A Guide to A/B Testing Tools. *EContent, 38*(7), 30–31. Retrieved from <https://www.proquest.com/trade-journals/guide-b-testing-tools/docview/1724512649/se-2?accountid=27468>
- Koning, R., Hasan, S., & Chatterji, A. (2022, 9). Experimentation and Start-

- up Performance: Evidence from A/B Testing. *Management Science*, 68(9), 6434–6453. doi: 10.1287/mnsc.2021.4209
- Larsen, N., Stallrich, J., Sengupta, S., Deng, A., Kohavi, R., & Stevens, N. T. (2023). Statistical Challenges in Online Controlled Experiments: A Review of A/B Testing Methodology. *American Statistician*. doi: 10.1080/00031305.2023.2257237
- Lynch, D. (2022a). *Split Help Center - Foundational concepts*. Retrieved May 15, 2024, from <https://help.split.io/hc/en-us/articles/9648555765133-Foundational-concepts>
- Lynch, D. (2022b). *Split Help Center - Setting up and using metrics*. Retrieved May 15, 2024, from <https://help.split.io/hc/en-us/articles/9652327065485-Setting-up-and-using-metrics>
- Lynch, D. (2023). *Split Help Center - SDK validation checklist*. Retrieved May 15, 2024, from <https://help.split.io/hc/en-us/articles/13998631077901-SDK-validation-checklist>
- Machmouchi, W., & Buscher, G. (2016, 7). Principles for the design of online A/B metrics. In *Sigir 2016 - proceedings of the 39th international acm sigir conference on research and development in information retrieval* (pp. 589–590). Association for Computing Machinery, Inc. doi: 10.1145/2911451.2926731
- Martin, E. J. (2015). The ABCs OF A/B TESTING. *EContent*, 38(7), 12–17.
- Nassi, T., & Jewkes, H. (2021). *SIMULTANEOUS EXPERIMENTATION: RUN MULTIPLE A/B TESTS CONCURRENTLY*. Retrieved June 5, 2024, from <https://www.split.io/blog/simultaneous-experiments/>
- Optimizely. (n.d.-a). *Compliance*. Retrieved May 15, 2024, from www.optimizely.com/trust-center/compliance/
- Optimizely. (n.d.-b). *FAQs*. Retrieved May 15, 2024, from <https://support.optimizely.com/hc/en-us/sections/4410282498061-FAQs>
- Optimizely. (n.d.-c). *How can we help you?* Retrieved May 15, 2024, from <https://support.optimizely.com/hc/en-us>
- Optimizely. (n.d.-d). *Optimizely developer documentation*. Retrieved May 15, 2024, from <https://docs.developers.optimizely.com/>
- Optimizely. (n.d.-e). *Plans pricing*. Retrieved May 15, 2024, from www.optimizely.com/plans/
- Optimizely. (n.d.-f). *Troubleshoot*. Retrieved May 15, 2024, from <https://support.optimizely.com/hc/en-us/sections/13836867450637-Troubleshoot>
- Optimizely. (2023a). *Create a user context using the JavaScript (Browser) SDK*. Retrieved May 15, 2024, from <https://docs.developers.optimizely.com/feature-experimentation/docs/create-user-context-javascript>
- Optimizely. (2023b). *JavaScript (Browser) SDK quickstart*. Retrieved May 15, 2024, from <https://docs.developers.optimizely.com/feature-experimentation/docs/javascript-browser-quickstart>
- Optimizely. (2023c). *QA checklist*. Retrieved May 15, 2024, from <https://docs.developers.optimizely.com/feature-experimentation/docs/qa>

[-checklist](#)

- Optimizely. (2023d). *Track Event for the JavaScript (Browser) SDK*. Retrieved May 15, 2024, from <https://docs.developers.optimizely.com/feature-experimentation/docs/track-event-javascript>
- Optimizely. (2023e). *Troubleshoot*. Retrieved May 15, 2024, from <https://docs.developers.optimizely.com/feature-experimentation/docs/troubleshoot>
- Optimizely. (2024a). *Example usage of the JavaScript (Browser) SDK*. Retrieved May 15, 2024, from <https://docs.developers.optimizely.com/feature-experimentation/docs/example-usage-javascript>
- Optimizely. (2024b). *FAQ*. Retrieved May 15, 2024, from <https://docs.developers.optimizely.com/feature-experimentation/docs/faq>
- Optimizely. (2024). *The home of exceptional digital experiences*. Retrieved May 15, 2024, from www.optimizely.com/
- Optimizely. (2024a). *Optimizely Experiment Results page*. Retrieved May 15, 2024, from <https://support.optimizely.com/hc/en-us/articles/4410284017421-Optimizely-Experiment-Results-page>
- Optimizely. (2024b). *Quickstarts*. Retrieved May 15, 2024, from <https://docs.developers.optimizely.com/feature-experimentation/docs/quickstarts>
- Optimizely. (2024c). *Run A/B tests*. Retrieved May 15, 2024, from <https://docs.developers.optimizely.com/feature-experimentation/docs/run-a-b-tests>
- Optimizely. (2024d). *Stats Accelerator*. Retrieved May 15, 2024, from <https://support.optimizely.com/hc/en-us/articles/4410283570701-Stats-Accelerator>
- Optimizely. (2024e). *Types of metrics and when to use them*. Retrieved May 15, 2024, from <https://support.optimizely.com/hc/en-us/articles/4410289437325-Types-of-metrics-and-when-to-use-them>
- Optimizely - Dev guide. (2024). *Introduction - Optimizely Feature Experimentation*. Retrieved May 15, 2024, from <https://docs.developers.optimizely.com/feature-experimentation/docs/introduction>
- Piwik PRO. (2024). *Integrations*. Retrieved May 15, 2024, from help.piwik.pro/support/integrations/
- Quin, F., Weyns, D., Galster, M., & Silva, C. C. (2024, 5). A/B testing: A systematic literature review. *Journal of Systems and Software*, 211. doi: 10.1016/j.jss.2024.112011
- Siroker, D., Koomen, P., & Harshman, C. (2013). *A/B testing the most powerful way to turn clicks into customers* (1st edition ed.). Hoboken: Wiley.
- Split. (n.d.-a). *About Split - WE POWER TEAMS TO MAKE AN IMPACT*. Retrieved May 15, 2024, from www.split.io/company/
- Split. (n.d.-b). *JOIN FREE AND SCALE FROM THERE*. Retrieved May 15, 2024, from www.split.io/pricing/
- Split. (n.d.-c). *Split*. Retrieved May 15, 2024, from <https://app.split.io/>

- Split. (n.d.-d). *Split Help Center - Client-side SDK examples*. Retrieved May 15, 2024, from <https://help.split.io/hc/en-us/sections/360004026891-Client-side-SDK-examples>
- Split. (n.d.-e). *Split Help Center - Server-side SDK examples*. Retrieved May 15, 2024, from <https://help.split.io/hc/en-us/sections/12619407921677-Server-side-SDK-examples>
- Split. (n.d.-f). *Split Help Center - Step-by-step*. Retrieved May 15, 2024, from <https://help.split.io/hc/en-us/sections/360003930711-Step-by-step>
- Split. (n.d.-g). *Trust Center*. Retrieved May 15, 2024, from <https://security.split.io/>
- Split. (2018a). *Split Help Center*. Retrieved May 15, 2024, from <https://help.split.io/hc/en-us>
- Split. (2018b). *Split Help Center - Edit treatments*. Retrieved May 15, 2024, from <https://help.split.io/hc/en-us/articles/360020525112-Edit-treatments>
- Split. (2018c). *Split Help Center - Events*. Retrieved May 15, 2024, from <https://help.split.io/hc/en-us/articles/360020585772-Events>
- Split. (2018d). *Split Help Center - JavaScript SDK*. Retrieved May 15, 2024, from <https://help.split.io/hc/en-us/articles/360020448791-JavaScript-SDK>
- Split. (2018e). *Split Help Center - Metrics impact tab*. Retrieved May 15, 2024, from <https://help.split.io/hc/en-us/articles/360020844451-Metrics-impact-tab>
- Split. (2018f). *Split Help Center - Traffic type*. Retrieved May 15, 2024, from <https://help.split.io/hc/en-us/articles/360019916311-Traffic-type>
- Split. (2018g). *Split Help Center - Understanding metric impact*. Retrieved May 15, 2024, from <https://help.split.io/hc/en-us/articles/360020890491-Understanding-metric-impact>
- Split. (2019a). *Split Help Center - Browser RUM Agent*. Retrieved May 15, 2024, from <https://help.split.io/hc/en-us/articles/360030898431-Browser-RUM-Agent>
- Split. (2019b). *Split Help Center - Metric details and trends*. Retrieved May 15, 2024, from <https://help.split.io/hc/en-us/articles/360025376251-Metric-details-and-trends>
- Split. (2019c). *Split Help Center - Naming event types*. Retrieved May 15, 2024, from <https://help.split.io/hc/en-us/articles/360023227391-Naming-event-types>
- Split. (2019d). *Split Help Center - Running an A/A test*. Retrieved May 15, 2024, from <https://help.split.io/hc/en-us/articles/360031279312-Running-an-A-A-test>
- Split. (2019e). *Split Help Center - SDK overview*. Retrieved May 15, 2024, from <https://help.split.io/hc/en-us/articles/360033557092-SDK-overview>

- Split. (2019f). *Split Help Center - Troubleshooting*. Retrieved May 15, 2024, from <https://help.split.io/hc/en-us/articles/360025918571-Troubleshooting>
- Split. (2020a). *Split Help Center - Split's approach to statistics*. Retrieved May 15, 2024, from <https://help.split.io/hc/en-us/articles/360042265892-Split-s-approach-to-statistics>
- Split. (2020b). *Split Help Center - Using dependencies to run mutually exclusive experiments*. Retrieved June 5, 2024, from <https://help.split.io/hc/en-us/articles/360038843991-Using-dependencies-to-run-mutually-exclusive-experiments>
- Split. (2021a). *Split Help Center - Browser SDK*. Retrieved May 15, 2024, from <https://help.split.io/hc/en-us/articles/360058730852-Browser-SDK>
- Split. (2021b). *Split Help Center - Constructing a hypothesis*. Retrieved May 15, 2024, from <https://help.split.io/hc/en-us/articles/360055681831-Constructing-a-hypothesis>
- Split. (2021c). *Split Help Center - Share results*. Retrieved May 15, 2024, from <https://help.split.io/hc/en-us/articles/360059696231-Share-results>
- Split. (2023). *Split API documentation - Introduction*. Retrieved May 15, 2024, from <https://docs.split.io/reference/introduction>
- Split. (2024a). *Feature experimentation - release, test learn as you go*. Retrieved May 15, 2024, from www.split.io/product/experimentation/
- Split. (2024b). *Feature Flags, Experimentation + Continuous Delivery*. Retrieved May 15, 2024, from www.split.io/
- Stuart, T. (2019a). *Split Help Center - Step 1: Install the SDK*. Retrieved May 15, 2024, from <https://help.split.io/hc/en-us/articles/360025334751-Step-1-Install-the-SDK>
- Stuart, T. (2019b). *Split Help Center - Step 2: Create a feature flag and target users*. Retrieved May 15, 2024, from <https://help.split.io/hc/en-us/articles/360025334851-Step-2-Create-a-feature-flag-and-target-users>
- Stuart, T. (2019c). *Split Help Center - Step 3: Send event data*. Retrieved May 15, 2024, from <https://help.split.io/hc/en-us/articles/360025335031-Step-3-Send-event-data>
- Vattenfall. (n.d.). *Oma energia -palvelu*. Retrieved May 17, 2024, from <https://www.vattenfall.fi/asiakaspalvelu/oma-energia-palvelu/>
- VWO. (2024a). *Digital Experience Optimization*. Retrieved May 15, 2024, from vwo.com/
- VWO. (2024b). *Feature toggles*. Retrieved May 15, 2024, from vwo.com/glossary/feature-toggles/
- VWO. (2024c). *Sdks supported in vwo fullstack*. Retrieved May 15, 2024, from help.vwo.com/hc/en-us/articles/900001640366-SDKs-Supported-in-VWO-FullStack

Zelaya, N. (2023). *Split Help Center - Browser Suite*. Retrieved May 15, 2024, from <https://help.split.io/hc/en-us/articles/22622277712781-Browser-Suite>

A A/A test results

AAtesting | Prod-Default

Powered by 

Feature flag version

v7 (Jun 6, 2024 09:24 PM GMT+3 to Jun 13, 2024 09:41 PM GMT+3)

With targeting rule

default rule

Comparison treatment



on (Unique users: 3,992)

Compared against baseline treatment

off (Unique users: 4,020)





Metric tag filters

Metric owner filters



Duration: 7 days 0 hours and 17 minutes |  Experimental period: Complete |  Sample ratio: Valid

Testing type: Fixed horizon

Key Metrics

FP card clicks: ConsumptionSummary	Statistically inconclusive	 1.38%	Impact lies between -17.07% and +14.31%
FP card clicks: ContractsInstruction	Statistically inconclusive	 4.41%	Impact lies between -28.61% and +19.78%
FP card unique users: ContractsInstruction	Statistically inconclusive	 10.63%	Impact lies between -32.63% and +11.37%
FP card unique users: ConsumptionSummary	Statistically inconclusive	 5.69%	Impact lies between -15.44% and +4.06%

Organizational Metrics

FP card clicks: SpotPrices	Significant desired impact	 79.72%	Impact lies between +3.69% and +155.76%
FP card clicks: ElectricityStatusIndicator	Statistically inconclusive	 4.42%	Impact lies between -100.62% and +91.79%

FP card clicks: ExpensesDonutChart	Statistically inconclusive	- 2.89%	Impact lies between -28.05% and +22.26%
FP card clicks: TimeOfDayInsight	Statistically inconclusive	+ 7.49%	Impact lies between -55.09% and +70.07%
FP card unique users: ElectricityStatusIndicator	Statistically inconclusive	- 19.44%	Impact lies between -100.00% and +66.35%
FP card unique users: ExpensesDonutChart	Statistically inconclusive	+ 5.72%	Impact lies between -16.85% and +28.29%
FP card unique users: SpotPrices	Statistically inconclusive	+ 19.08%	Impact lies between -16.63% and +54.80%
FP card clicks: FenniaAd	Metric not available	Metric not available as the calculation did not re- turn. Our support engineering team has been noti- fied.	
FP card clicks: Insight	Metric not available	Metric not available as the calculation did not re- turn. Our support engineering team has been noti- fied.	
FP card unique users: FenniaAd	Metric not available	Metric not available as the calculation did not re- turn. Our support engineering team has been noti- fied.	

Experiment settings

Significance threshold 0.05	Minimum sample size 355	Experimental review period 7 Days
Multiple comparison correction Applied		
Organization name Helen	Workspace name OmaHelen	Report generated on Jun 13, 2024 10:46 PM GMT+3

[View in Split](#) 

B A/B test results

SwitchCardPlacement | Prod-Default

Powered by 

Feature flag version

v5 (Jun 5, 2024 12:56 PM GMT+3 to Jun 12, 2024 12:59 PM GMT+3)

With targeting rule

default rule

Comparison treatment



on (Unique users: 3,976)

Compared against baseline treatment

off (Unique users: 3,984)





Metric tag filters

Metric owner filters



Duration: 7 days 0 hours and 3 minutes |  Experimental period: Complete |  Sample ratio: Valid

Testing type: Fixed horizon

Key Metrics

FP card clicks: ContractsInstruction	Significant desired impact	 54.78%	Impact lies between +26.45% and +83.11%
FP card unique users: ContractsInstruction	Significant desired impact	 48.89%	Impact lies between +22.50% and +75.27%
FP card clicks: ConsumptionSummary	Statistically inconclusive	 12.04%	Impact lies between -25.22% and +1.14%
FP card unique users: ConsumptionSummary	Statistically inconclusive	 1.24%	Impact lies between -10.15% and +7.67%

Organizational Metrics

FP card clicks: ExpensesDonutChart	Significant undesired impact	 21.10%	Impact lies between -41.27% and -0.94%
FP card unique users: ExpensesDonutChart	Significant undesired impact	 18.21%	Impact lies between -35.94% and -0.48%

FP card clicks: ElectricityStatusIndicator	Statistically inconclusive	+ 0.00%	Impact lies between -101.42% and +101.42%
FP card clicks: SpotPrices	Statistically inconclusive	- 20.60%	Impact lies between -65.15% and +23.95%
FP card clicks: TimeOfDayInsight	Statistically inconclusive	- 20.22%	Impact lies between -69.12% and +28.68%
FP card unique users: ElectricityStatusIndicator	Statistically inconclusive	+ 23.32%	Impact lies between -85.79% and +132.44%
FP card unique users: SpotPrices	Statistically inconclusive	- 25.01%	Impact lies between -51.61% and +1.58%
FP card clicks: FenniaAd	Metric not available	Metric not available as the calculation did not return. Our support engineering team has been notified.	
FP card clicks: Insight	Metric not available	Metric not available as the calculation did not return. Our support engineering team has been notified.	
FP card unique users: FenniaAd	Metric not available	Metric not available as the calculation did not return. Our support engineering team has been notified.	

Experiment settings

Significance threshold 0.05	Minimum sample size 355	Experimental review period 7 Days
Multiple comparison correction Applied		
Organization name Helen	Workspace name OmaHelen	Report generated on Jun 12, 2024 01:00 PM GMT+3

[View in Split](#) 

Page 2

C Quick guide to A/B testing

1. Define purpose of the service or aspects that need improvement
 - Based on the goals of the service long-term metrics can be determined
 - For instance, if there is a goal to improve the use of self-service functionalities, the use of them should be tracked
 - In addition to long-term metrics, metrics specific to the problem/feature can be identified and tracked as well as debug/guardrail metrics can be used (e.g. page load times,..)
2. Identify bottlenecks in the service
3. Construct hypothesis based on the knowledge on the users
 - A hypothesis makes the tests more informative by displaying what the test aims to determine
 - Template: *Because we observed X, we expect that by doing Y it will cause users to Z.*
4. Create the new variation(s) based on the hypothesis
5. Run the test
6. Analyse results
 - Is the new variant better/ worse or are the results same for both variants
7. Based on the results, another A/B test may need to be conducted, thus beginning the process from step 1

D Running an A/B test with Split

1. Create a feature flag which controls the variants (off is control and on is treatment variant)
2. Check the available metric on Metrics tab:
 - If there is already metrics created that measure suitable behaviour move to the next step
 - If there are no suitable metric(s):
 - Check if event(s) are tracked that can be used to calculate the metric
 - * If yes, move to creating new metric using the event
 - * If not implement event tracking (with `trackEvent` function) to the code
 - New metric is created on Split application:
 - * Go to Metrics
 - * Click “Create Metric”
 - * Add metric name and optionally description
 - * Select desired impact (do you want the value of the metric to increase or decrease)
 - * Select “user” as a traffic type (currently no other traffic types are available)
 - * Select how the metric is calculated
 - * Select which event(s) the metric uses for calculation
 - * Optionally, a filter or a cap can be set to the metric
3. After the changes are in production go to feature flag’s page on Split
4. Set Targeting rule to distribute treatment as follows (50-50)
5. The test should be now running (you may need to select which treatment is used as the baseline in metrics impact tab)
6. The results of the test are showed in the metrics impact tab
7. Allow the test to run at least 7 days, but additional 7 days may be needed if there is not enough data
8. See the meaning of the results from [here](#)

E What is A/B testing?

- The process of comparing two variants of the software (control and treatments) to determine whether there is difference between the variants
 - Users of the service are split into two halves and each half is served either the control or the treatment variant
 - The behaviour of the users is tracked and converted to metrics e.g. clicks
 - By comparing the metric values the difference can be calculated
- Allows determining whether a new feature is improving the service
- Allows tracking long-term goals of the service
- Can be considered as user testing, but does not require as much resources as general user testing
- Can be applied in mobile, front-end or back-end
- Can be used to test for instance:
 - Visual elements
 - Algorithms
 - Workflows
- Benefits:
 - Mitigates risks
 - Ensures product quality
 - Allows discovering and validating value
 - Can stabilise and lower product complexity
 - By adopting A/B testing as part of the workflow it can increase performance by 30 to 100 %