Hypofund: an online platform for personalised mortgage advice

Henk Bierlee
Delft University of Technology

Kilian Grashoff
Delft University of Technology

TU Coach: Annibale Panichella

Bachelor Project Coordinators:
Felienne Hermans
Otto Visser
Martha Larson

June 29, 2016

Foreword and acknowledgements

This report was created by Kilian Grashoff and Henk Bierlee as the conclusion of the TI3806 Bachelor project course of the Bachelor Computer Science at the Delft University of Technology. It contains a comprehensive overview of the research, specifications, implementation, evaluation, and discussion we have conducted over the past ten weeks.

We would like to thank our client and technical coach at Hypofund for their excellent support and enthusiasm during the project. Being part of their start-up at this stage was a truly educational experience.

We would also like to thank our TU Coach Annibale Panichella of the Software Engineering Research Group for all the advice and help he extended to us throughout the process. The weekly meetings with him in Delft and his quick and useful responses, feedback and criticism helped us through the harder parts of the academic side of the project.

Finally, we would like to thank the bachelor project coordinators Felienne Hermans, Otto Visser and Martha Larson for organising the course and helping us with practical arrangements for this project.

Contents

Summary

This is a project report on the research, implementation and empirical test of an online system for mortgage advice that have been completed in the context of our bachelor project. The project objective was to build a platform for online mortgage advice, which could substitute a real-life, human adviser. The use of such a platform costs far less than hiring a human adviser, but a major problem to solve is that a website usually cannot give personal treatment to the user.

In the research phase, we looked at how to apply the fields of affective and adaptive interfaces to our problem domain, as well as how A/B testing is commonly done for software products and how the results are statistically analyzed. With the knowledge we gained from the research, we came to an agreement of what to build with the client. The result of this agreement is the specification of the 'hybrid workflow' interface (tailored to the philosophy of the client's vision), a personalisation feature and an empirical test. We implemented said specification and conducted a proof-of-concept A/B test with 47 users.

After statistical analysis, we concluded that the results of these tests were not significant: there was no significant difference between the users that got a variant of the platform with the personalisation feature and the users that got the variant without our feature. In the discussion segments, we discuss the results and make further recommendations.

1 Introduction

During the last decades, the web has been growing more accessible and mainstream every year. This has led to web-platforms being used for services that were traditionally performed by humans. Among the greatest challenge companies building these web-platforms are faced with is substituting the human element. In our context, this means we are trying to provide the personal touch a mortgage adviser brings to conversations with their clients. It is one thing to set up a website that can calculate some mortgage numbers, but it is another to make a user feel that they are receiving a personal treatment.

1.1 Problem statement

Our client was the head of a startup company at a very early stage. The working title of the startup is Hypofund. The end-product is an online mortgage advice tool, which allows users to close a mortgage on the site, saving them the expense of hiring a real-life mortgage adviser.

Our project has three components. The first is to develop a prototype for a core part of the minimal viable product, namely the interface where the user inputs their information and receives mortgage advice. The second component was to research and implement an innovative way to make the experience of the user more personal, in order to compete with real-life mortgage advisers. If our prototype increases the conversion rate, it could be incorporated in the Hypofund end-product. This added the third part to the problem: evaluating the personalisation, for which we developed a user test.

1.2 Overview of the report

In this bachelor project, we have researched, implemented and empirically evaluated a web-platform which adapts to the personality of the user. In section 2, the results of the research phase are presented. In section 3, we analyse the problem, talk about the considered solutions and state what was agreed to build. This section is split into the three components of the project, which are the interface, personalisation and A/B test. Next, in the implementation report in section 4, we give a general and sprint by sprint implementation report. In section 5, we evaluate the results of the A/B test and draw conclusions on the results. In section 6, we discus the evaluation conclusions, the research and reflect on the process. Finally, in section 7 we draw our overall conclusions about the project.

2 Research

In this section we present the research we have done for this project and on which the product is based. The research topics are affective interfaces, adaptive interfaces and A/B testing.

In the broadest sense, the problem to be solved was to replace a real-life mortgage advisor with a web-based tool, while sacrificing as little as possible of the personal touch and adaptiveness of a human interaction. A more extensive report on the client needs and requirements can be found in the sections after research. We were expected to come up with our own creative solutions on how to achieve this, so we started our research by getting acquainted with general topics of human-computer interaction.

During the research phase, we frequently met with the client to talk about the product and discuss proposals for how to solve the problem. In this manner, we narrowed down research topics or found new directions to look for interesting approaches. This is how we came to affective interfaces, and later adaptive interfaces.

2.1 Affective interfaces

Human to human interaction is a dynamic and complex process. While humans have been trained all their life to adapt to the emotions of other humans and do it mostly subconsciously, teaching computers to pick up on subtle clues is an entirely different case. A person's affect, which is the experience of feeling emotion, is displayed only through very subtle clues. Sometimes these clues are explicit, such as a person saying 'I'm angry' or 'I'm sad', but they can also be implicitly conveyed through tone of voice, facial expression or body language. [23]

2.1.1 The need for affective interfaces

So-called 'affective interfaces' are the result of the effort to find ways to effectively factor in the user's affect into the interface. The first case for affective interfaces was made in 1995, by Picard, in her paper Affective Computing. [23]

In this paper, it is argued that emotions play an important role in many neurological processes. This means that for computers to fulfill their potential when it comes to interacting with humans, they will necessarily have to be affective as well. Section 1.4 reports an example of a 'piano teaching computer system'. Without the ability to recognize when the user is frustrated (which usually leads to quitting), the computer can never be as effective at teaching piano compared to a human teacher. This situation is analogous to ours, where we want to replace a human professional with a computer system.

2.1.2 Ways to measure affect

There are many pitfalls when it comes to interfaces that adapt based on affect. When the interface adapts to the user based on its perception of the user's emotional state, there is the risk that the interface becomes unpredictable for the user. Even worse is when the changes is unhelpful, and the interface becomes less user friendly. These changes can induce frustration. This is why there

should be systems in place that determine the helpfulness of any changes in the interface, for example by measuring frustration of the user. With this feedback loop, changes in the interface can be reverted when the computer misperceives the user's emotional state.

In study by Reynolds and Picard [26], such a feedback loop was implemented via experience sampling, but also via specially build hardware, such as a pressure sensitive mouse. The idea is that the pressure the user applies to mouse is an indicator of frustration. The study was small, but feedback was positive. This study illustrates the point that when building an interface that changes in response to measurements about the user, it is important to evaluate the changes the adaption of the interface causes.

It also illustrates that one of the biggest challenges in affective computing is how to let computers measure or sense the user's emotional state, and which input to take from the user. The prototype of the pressure sensitive mouse works well, and users reported liking the concept. Another study [29] also used a pressure sensitive mouse, but unlike the mouse from the original experiment, the sensors on the prototype were hidden. The study was also larger, with 144 users. This study confirmed that the assessment of arousal via the registration of force on the mouse buttons is reliable. In practice, the obvious problem with specialized hardware like pressure sensitive mice is that they are not commercially available, and therefore not commonly owned by users. There is no indication that this will change soon.

One option that has been explored in a previous study by Polzin and al. [24], was to use the user's voice, which after all is also used by humans to gauge each other's emotions. In this study, a model was trained on voice samples from a database of movies. The voice segments were tagged as either 'neutral', 'sad' and 'angry', and the results from different classifiers were compared to the results of a control group of humans. The experiment was moderately successful: the best classifier achieved an accuracy of 60%, while the humans achieved an accuracy of 70%.

Relying on voice input is an interesting direction, however, 60% is not very reliable. The other problem with voice input is that we'd have to request permission to activate the user's microphone, which is not realistic as most users would not agree to this. So, we agreed with the client pursue this method.

2.1.3 Other research in affective design

Applications that implement affective design do not have to explicitly measure and dynamically adapt to the user to account for the user's affect. Taking into account how the user will respond emotionally to specific design choices in the design of a website can also be effective in creating a better interaction experience for the user.

One paper by Egger [14] did extensive research on how to raise the user's level of trust in an e-commerce website. It presents a Model of Trust for E-Commerce (or MoTEC), a framework that provides design principles and concrete guidelines for websites that sell products and services over the internet. In this framework, user concerns with the site are systematically addressed. For us, this was valuable research, because if our product clashes with the guidelines of MoTEC, chances are that the negative effects from this would overshadow the benefits of the feature.

For example, the guideline to 'let the user be in control' can clash with adaptive interfaces. If an interface adapts automatically outside the control of the user, it can be frustrating for the user if the change is not helpful. This negative effect is further aggravated since the user did not directly make the change happen themselves: this often means that reverting the change is even more difficult, because the user does not know where to find the option to revert the change.

2.2 Adaptive interface design

Because most techniques used in Affective design did not seem suited for our product, because we were constrained in that we only had use of the data provided by a modern web browser. Because most affective interface research has focused on using specialized hardware and input data (such as voice and video), we switched to studying adaptive interfaces in general. An adaptive interface is defined as an interface that changes its layout and elements to the needs of the user or context. [30]

2.2.1 Variants of adaptive interface design

First, the variants of interface adaption that are described in existing literature were researched. This is important, since we have to know what types of adaptation are possible in order to choose the option best suited for Hypofund.

Types of interface customization Rossi et al. define three types of interface customization: link customization, structure customization, and node customization. [28]

When using link customization, hyperlinks on a webpage are customized for each user, to provide them with links to pages that are most relevant to them. Link customization often takes the form of a "recommended for you" block on the page, or similar constructs. This is useful, since it is clear to the user which links change, and which ones do not.

Secondly, structure customization means that the structure of a page changes. So, while the elements on the page remain the same, groups of elements are moved around.

Finally, node customization involves swapping the content on a page for different content. Of all the types of customization, this is the most invasive, since it is the only one that changes the main content of the page. This means that node customization both provides the most freedom, but is also the most difficult to implement well.

We determined that for us, node customization is the most applicable. Since the our interface is a single page app, link customization is not applicable. Structure customization is difficult to do, since many steps in the user's workflow have to be completed in a certain order. Therefore node customization remains.

Types of user and context profiling Besides the types of customization described above, there are also several ways in which to build the profile that is used as input data for the adaptation.

Partarakis et al. describe two types of profile that can be built. [22] The goal of user profiling is to: "provide information regarding the user who accesses

an interactive application. A user profile contains attributes either specified by the user prior to the initiation of interaction or acquired by the system during interaction (through interaction monitoring)." Context profiling aims to collect properties of the machine and environment that are unlikely to change during interaction. [22]

Since we are interested in providing the user with a personal experience, it is most likely that we will use user profiling. However, context profiling may also be interesting, because the context can also provide (indirect) information about the user.

For both user and context profiling, Partarakis et al. describe two ways in which this profile can be built. [22] In static profiling the properties that will be collected about the user or context, are entirely specified beforehand. Extensible profiling aims to separate the logic for adaptation from the specification of properties that will be collected. This allows extra properties to be added to the user or context profile while the system is running, without having to alter the adaptation logic itself. [22]

While extensible profiling is a more dynamic and generally applicable system, we will be using static profiling, since this keeps our architecture more light-weight. We do not think the benefits of extensible profiling outweigh this extra development cost for our client, since the client is only interested in applying adaptations that are shown beforehand to be clearly beneficial to the user experience. Secondly, we suspect the type of customization that we will perform will be closely tied to the properties that the customization relies on from the user profile, which reduces the benefit from being able to specify logic and properties separately.

2.2.2 Architectural considerations

Even using only static profiling, adaptive interface can pose architectural challenges. Rossi et al "strongly claim that [they] can build high quality personalized software only if [they] design that software with flexibility and extension in mind from the beginning. [28]

There are different parts of an application where one can choose to implement the adaption, and in order to evaluate the options carefully we looked at various frameworks that try to help structure the way in which adaptation is implemented.

The OOHDM Design Approach Object-Oriented Hypermedia Design Method (OOHDM) is a model-based approach that is used to develop web applications. Rossi et al. explain that "the key concept in OOHDM is that Web application models involve a Conceptual, a Navigational Model and an Interface Model. Those models are built using object-oriented primitives with syntax close to UML." [28]

The conceptual model defines "domain objects, relationships and the intended applications' functionality." Nodes specify domain objects or classes implementing functionality. Relations specify how the various data types and classes relate conceptually to each other, similar to class diagrams in UML. The elements of this model are usually not affected by adaptation, since whatever data the user is shown, the types of data the system uses does not change. However, the conceptual model does play a role in personalisation, since the

users, roles and groups that are present in the personalisation system should be explicitly specified in the conceptual model.

The elements of the navigational model (both nodes and relations) are defined as views on elements in the conceptual model. The term view in this context means a way to look at the underlying data, in this case the conceptual model. The relations in the navigational model represent how the user can navigate from one element to another.

When doing structural customization, the relations in the navigational model are adapted for individual users. When doing node customization, the contents of nodes are adapted. This means that for both structural and node customization, the navigational model is adapted to the user or context.

Finally, the interface model specifies the look and feel of navigation objects. This allows several interfaces, like a web-interface, a desktop application and a mobile app to be defined separately from the navigational model. The interface model is usually not involved in personalisation, since it should be independent from all functionality. Since we are only building a web-interface, we will have only one interface model.

Layer model for context-aware systems Context-awareness is defined as "the ability of a service, application or actuator to adapt to a specific context. [17] The various layers for a context-aware middleware are (from bottom to top):

1. Acquisition: the collection of relevant (sensor) data
2. Reasoning: deducing meaningful information from the data
3. Management of context information
4. Actuation: adapting application behavior to the available context information [17] (as cited in [12])

In this model, each layer only has communicates with the layers above and below it. Each layer provides data to the layer above, and uses data from the layer below. The first and fourth layer are exceptions: the acquisition layer gets its data from user input or from data collected about the user in another fashion. The fourth layer results in changes to the interface.

Even though we are not building a context profile, but a user profile, the same layer model can be applied. The only difference between the two is in the acquisition layer, where we directly collect data about the user, instead of about the user's context. By separating our system into these layers, there is a clear separation of concerns.

Comparison with OOHDM The OOHDM Design Approach and layer model for context-aware systems serve two different purposes.

The OOHDM Design Approach helps to structure parts of the system by separating the interface, navigation and conceptual models, in a fashion similar to the MVC [16] approach. It specifies a modeling language to build these different models, which can then inform the structure of the application.

Although thinking about the various elements in our system using the OOHDM design approach was very beneficial for us, we decided against using its modeling language to specify the design of our system. The reason for this is that we

considered it to be too heavyweight for our relatively simple architecture using only static profiling. Also, the approach partially addresses the same concerns as an MVC-like architecture as used by the Django web-framework we used. Since using this framework already forced us to structure our application in an MVC fashion, we did not think that OOHDM would provide a substantial benefit to us.

On the other hand, the layer model for context-aware systems directly specifies a clear model in which the application code can be structured, without having to specify a model using diagrams beforehand. For each functionality of the system, one can determine in which of the four layers it fits best, and implement it in that module. Because the layer model is specifically tailored to our type of system, we think it will fit the functionality we will implement well.

2.3 A/B testing

In software, a common way to demonstrate that changes have a (positive) impact is to do user testing. In software development, A/B testing is used by many companies, such as Amazon, to make informed and data-driven decisions about design, functionality and most everything else. [20] This is no wonder, since A/B testing is a tool that finds causal relationships with a high probability [19]

The model also lends itself well to the influential lean startup methodology, established by Eric Ries, which fits the context of our client and project. From an excerpt from his book on the lean startup methodology:

"[A/B testing] often uncovers surprising things. For example, many features that make the product better in the eyes of engineers and designers have no impact on customer behavior." [27]

In other words, even if we as the designers and engineers rationalize the benefits of our personalisation feature, the outcome of the A/B test shows the bottom truth: the impact on the user.

2.3.1 A/B testing in general

Kohavi et al. wrote a paper as a practical guide and case study of A/B testing, which we have used as guiding principles in setting up our own A/B test. [18]

According to this paper, A/B testing is a randomized experiment where some of the available test subjects (the test users) are served one variant of the product, and others another variant. Typically, there is only one element that is different between the two variants, of which the researchers want to evaluate and quantify the effects. For instance, the first variant might be the current site, and the second variant the site with an additional feature, or a different design. For every user, one or more test metrics are collected, such as conversion rate, completion time or an opinion survey. At the end of the test, the variant that scored significantly better can be implemented.

A/B testing can be done for products while they are live and in production. [18] For companies that have established a substantial user-base already, there are several advantages:

1. An A/B test can be deployed for a small portion of the users, without the users even being aware that they are in a test. A test user in a laboratory does not act as naturally.

2. The users are using the product out of their own volition, not because of an external motivation (such as a compensation for participating in a user test)

3. Taking Facebook as an example: most users that are on Facebook are target users for Facebook

All the metrics (both objective workflow measurements and the subjective feedback from the evaluation survey) are combined in the Overall Evaluation Criterion, or OEC. This is the quantitative measure of the experiment's objective and should be decided before any data is interpreted. [18]

2.3.2 Statistical analysis of OEC

For the statistical analysis, we used our knowledge of statistical techniques from the bachelor subject TI2216M Kansrekening en Statistiek which used the book A modern introduction to probability and statistics. [13] We also conferred with our TU coach to determine what statistical tests would be most applicable to use to evaluate the results of the A/B test. Those statistical tests were then researched further to understand how to apply them properly.

Statistical principles Before any tests are performed, it is important to determine the confidence level that will be used for all tests. For every test we establish the Null hypothesis, the Test hypothesis and what statistical test to use. If the statistical test results in a p value lower than the p value for our confidence level, the Null hypothesis is rejected.

The Null Hypothesis is the hypothesis that there is no significant difference between the variants and that any differences we observe are due to chance. For example, the primary Null Hypothesis of our research is that our personalisation feature has no effect on the user. This means that the goal of the project is to build an personalisation feature which has a strong enough impact, so we can reject the null hypothesis.

Even if our test shows a significant difference between control and test OECs, there is still a probability that the Null Hypothesis is true. This probability is called the confidence level, which we set to the commonly used value of 95%.

When the Null Hypothesis is false, there is also a probability that this fact will not be revealed by our test. The probability with which our test detects that there is a difference when it is actually there, is called the Power.

Normality test To determine whether the OEC and other resulting data for individual groups of samples is normally distributed we used the Shapiro-Wilk test for normality. This is the most powerful normality test for low samples sizes ($n < 50$) of the various normality tests that are frequently used. [25] [21]

It should be noted that all normality tests tend to have little power for small sample sizes, so they should not be relied on to reject the null hypothesis {H0: the samples under test are normally distributed} for small sample sizes. This means that using a normality test to prove that another statistical test which relies on normality can be used for small sample sizes can be deceptive, since the normality test may fail to reject H0 even though the samples under test are actually not normally distributed.

Despite the limited power for small sample sizes, using normality tests was useful for us. The reason for this is that a rejection of the null hypothesis is a strong indication that the samples are unlikely to be normal distributed, since the type-1 error rate of the test remains low, even for low sample sizes. [21] Our hypothesis is that the ratings within a group of users should likely be normal when there are no confounders that affect the group, and the ratings are not towards extreme ends of the scale (where a skew normal distribution is more likely). Therefore, a rejection of the null-hypothesis for the OEC indicates that there may be confounders affecting the OEC within the tested group of users, warranting further investigation with statistical tests for confounders. The results of the normality tests were used for this purpose, but are not listed in this report, since they are not relevant to our conclusions.

Tests for likelihood of population difference A statistical test for population difference tests the null hypothesis that two samples come from the same population against the alternative hypothesis that they came from the same population. We cannot assume that our samples are normally distributed, regardless of what the normality test says about our samples, due to the reasons described in the previous paragraph. For this reason we chose to use only non-parametric tests, which do not assume that the samples are normally distributed.

To test our test sample versus our control sample, we used the Mann–Whitney–Wilcoxon test. This test accepts two samples as inputs, and returns a probability that the samples came from the same population without assuming a distribution. We did not use a continuity correction, since the ratings our users are allowed to give are discrete.

Test for confounding variables To test for possible confounding variables, we split the test and control group based on the possible confounder, and used the Kruskal–Wallis one-way analysis of variance. This test accepts any number of samples as input, and returns the probability that they all come from the same population without assuming a distribution. If the test returns a low p-value it does not indicate which of the groups are probable to come from a different population. In that case, Dunn's test can be used to analyze which of the groups are different.

3 Agreement on what to build

In this section, we present what we proposed to build and what our rationale was behind the design of the three different components.

3.1 Interface

One of the central components of the project was to build a fully functional user interface, which would be dynamically personalised and evaluated. Many different requirements of this interface had to be satisfied for the project to be successful. Some were important, general requirements, which are described here. We used these to create the design of the interface.

Once we created the design, we derived a list of concrete requirement which had to be fulfilled to build the interface. This list can be found in Appendix A. In this section we present the challenges of the interface, from a technical and interaction design standpoint.

3.1.1 High-level goals and requirements

Firstly, the interface should accommodate the personalisation. Since personalisation is based on variations in the content, it should be possible to have multiple variants of the workflow available. The variant is stored in the database, hidden from the user: the user cannot know the variant, because it is a double blind test.

Secondly, the client has requirements from both a technical and a marketing standpoint. The technical requirements are presented in the implementation section. From a marketing and interaction design perspective, the client requires an interface that is in line with the Hypofund philosophy and the consumer target group, but we have free hand in how to achieve this. The long term goal of the Hypofund product is for customers to gain a broad financial insight, not necessarily pertaining to closing a one time mortgage. From meetings with the client, we have established a list of goals for the interface, the most important of which are:

- The product should give an overview of the user's financial situations. This should encourage revisits, when changes occur which impact the user's financial situation.
- No more information should be asked of the user than they are comfortable with
- The user should be encouraged to create a account on the site, however, the average user has many accounts already and has grown weary of creating even more
- The user should feel 'at home' at the Hypofund website

To gather accurate test results, the test-users have to feel like they are interacting with a finished, real-world product: a part of a website they actually might encounter online. This meant that the interface had to be a polished whole. To make this as feasible as possible in limited timeframe of the project, we opted for a user interface with a minimal, mostly standard design.

3.1.2 The workflow

The workflow is a user interface where the user inputs their information and which returns the result, such as maximum mortgage amounts and mortgage offers. During the design process, we looked at how competitors structured their workflow.

Existing workflows During the design process, we found that other mortgage advice websites commit to one of three distinct approaches to the workflow.

The first one is the step-by-step approach, which breaks up the process of entering user data into steps. The result is a direct and minimal workflow, which guides the user to the end-goal. Although there are usually options for the user to save his or her progress, the experience is based on a one-time runthrough without giving much reason for the user to return a second time after a mortgage deal was made.

The second was the more free-form profile workflow, where the user has the freedom to choose the order in which he inputs information. However, since some steps are dependent on others, the order is often still somewhat constrained.

A third option is the tool-based workflow, which is preferred by professionals because of its compactness. In this approach all the important input forms are put together on one page, where the resulting information is returned to the user. The downside is that the tool is more difficult to understand for inexperienced users.

The hybrid workflow Our interface design, which we coined the 'hybrid workflow', combines the first two approaches to create a hybrid of the step-by-step and profile workflows. The user inputs information step-by-step, but the interface clearly signals that the user is building a profile along the way, which they can check at all times. A summary of the profile with the most important data is permanently visible on the screen. The full profile is accessible at any time, and can be saved at any time. The profile is initially stored in the user's session cookie, but can also be permanently saved by the user. The important distinction with many other profile based websites, is to let the user save their work that is already there, instead of asking the user to register and invest beforehand.

In the hybrid workflow, a natural language dialogue between the user and the interface is established. Every step is split into panels that inform and engage the user, by providing introductory texts or by reacting on what the user has filled in thus far.

3.1.3 Landingpage

Initially, we also agreed to build a landingpage with an interactive mortgage calculator element. On this landingpage, the user could input some information and get an immediate estimate of their maximum mortgage. This functionality was fully designed and implemented, but was not included in the test as not to distract from the workflow.

3.2 Personalisation

Since one of the goals of Hypofund is to compete with real-life mortgage advisors, we looked at the main differences between real-life mortgage advisors and online mortgage advice solutions. We determined that possibly the biggest difference is that mortgage advisors adjust their approach to the personal preferences and personality of the user. A typical example of this is that a mortgage advisor will usually start by asking a client some questions about who they are, instead of diving right into the financial details. By doing this, the mortgage advisor can adjust the way in which they provide their advice to the client, and thereby give them better advice. Because all the other online mortgage advice solutions that we looked at just provide one possible workflow for users, they do not provide this level of personalisation.

Together with the client we decided to try to find ways in which we could personalise Hypofund for individual users. The following sections describe the process we went through to plan, specify and build a way to personalise Hypofund.

3.2.1 High-level goals and requirements

The main goal is to provide users with a personal experience. This means that not only should the user experience be improved by personalisation, but the user should also respond positively to the personalisation. If we improve the user experience itself, but the personalisation is annoying to the user for some other reason, the net effect may be negative.

From this goal we derived the following requirements for the personalisation feature:

- The user experience should be improved
- As many users as possible should complete the workflow. If we provide users with a great experience, but it is so complex that nobody completes it, there will be no value to the feature for our client. This leads to the following sub-requirements:
- If there are any added steps due to this feature, we should make sure that they are limited in time and effort required, so they do not cause users to quit
- The personalisation should never cause choices to become less clear to the user
- The personalisation should try to mimic a natural dialogue between two people, where one person tries to adapt to the other person's interests and preferences
- Compared to an offline mortgage advisor: when an advisor just gives you a pre-learned explanation without listening to you, this comes across as uninterested. In contrast, when an advisor adapts to what you tell them, this gives you the feeling that you are being listened to and getting an advice specifically tailored to you.
- The feature should not invade the user's privacy

- We should ask as little personal information from the user as possible
- It should always be clear to the user why we ask certain information from them, and why giving the information will lead to better advice

Many of the requirements above are highly subjective, and therefore difficult to evaluate. The section on A/B testing will provide more information on how we tried to measure the success of the personalisation feature, despite these difficulties.

3.2.2 Potential techniques for the personalisation feature

As we described in the research section (2) there are four main layers to a system for personalisation: [17]

1. Acquisition: gather input data which can be used to gain insight about the user
2. Reasoning: convert the input data to a metric that says something about the user and can be used to derive changes to the interface from
3. Management: store and manage the acquired information and metrics
4. Actuation: adapt the interface based on the model

For management, we will be using the Object-relational Mapping (ORM) features of the Django web-framework we will be using, as specified by our client. Using the goals and requirements specified in the previous section, we came up with the following possible techniques that could be used for the other three layers of a personalisation feature:

Acquisition

- Time a user takes to complete one step in the workflow where data is entered
- One example of this is to use completion time as a proxy for reading speed
- Mouse movement and clicks on the page
- Data that a user enters in forms
- Facial and audio recognition
- Typing speed
- Meta data, such as user agent string and referral URL

Reasoning metrics

- Frustration
- Doubt
- Taking time versus being in a rush
- Preference for longer detailed explanation versus succinct explanation
- Skill with computers
- Knowledge about mortgages

Actuation

- Structure: [28] choose to show to hide or show certain elements on the page for different groups of users
- Content: [28]
- Vary length of description text
- Vary style in which descriptive text is written
- Change other content, such as images that are shown
- Change the type of controls (like toggle buttons instead of checkboxes) for different users
- Design:
- Change used colors for interface elements
- Change used font
- Order of steps: the order of steps could be different for certain users. Some types of users might for instance be hesitant to give out information about their income until it is absolutely necessary.
- Refer users to customer support when they are having difficulties with the system.

Although we think all these possible techniques could be interesting, many are less feasible for various reasons. Some sensing techniques, like facial and speech recognition are technically hard to do in a browser and have privacy issues. Asking users to turn on their mic or webcam without giving a good reason is also practically not feasible. Many metrics are difficult to normalize properly without having prior data about the user. Finally, for many possible adaptations it is difficult to determine beforehand whether they will have the intended effect, and if implemented improperly, could backfire and annoy the user.

We went over these possible techniques with the client, to determine which ones seemed the most feasible, in order to arrive at a plan for the personalisation feature.

3.2.3 Proposed features for personalisation

By listing all the techniques and crossing off the ones that were unfeasible, we arrived at the following list of concrete personalisation features, which we proposed to the client and TU coach.

1. Use the referral URL to classify users into groups (using a rule-based approach, combined with targeted advertisements) and provide different landing and workflow pages for those groups. Since not all users can be classified, there needs to be a neutral option without personalisation. Possibilities of personalisation include:

   (a) Content of the steps

      i. Adjusting explanations and wording
      ii. Adding/removing specific questions per group

   (a) Prominence of the profile page (some users might not want a profile at all, while others might want to return to it after every step)
   (b) Design and content of the landing page
   (c) Design of workflow page

2. For users where the referral does not provide a clear group, add a step or questions to the personal profile to group people, and provide the same personalisation possibilities as 1.

   (a) Downside: this adds extra steps for the user, which could cause them to leave.
   (b) Upside: neutral option is no longer needed.

1. Detect user behaviour in the interface, and adjust content to fit their behaviour

   - Example: when a user clicks through after explanation text very quickly, this suggests that they did not read the entire text. An alternative shorter explanation for these users could be provided.
   - Upside: directly adapt to user preferences instead of indirectly
   - Downside: high risk of wrong classification, which can be annoying. Therefore difficult to implement correctly.

3.2.4 Accepted personalisation feature

Together with the client, it was determined that option 1a and 2 were the most interesting, since these seemed the most realistic to implement, while at the same time providing the most potential value to the client. Therefore, our final plan is a combination of 1a and 2: we adapt the content of the page to the personality of the user. We have two versions of the content, which we call customer journeys.

A user is assigned one of the two journeys based on their personality. Their personality is determined by either of the following tests:

1. The referral link. In practice, this referral can tell us from which advertisement the user came. Different advertisements are targeted to different personality types of users.
2. A personality test integrated in the system

For the user test we simulated a referral link with an external personality test, since this was easier than integrating the test into the system.

This plan fits well with the goals and requirements for the personalisation feature, since it provides a more personal dialogue with the user, which we think will improve user experience. An important aspect of the plan is that people should not be obliged to make the personality test in the final product, since this would introduce too many extra steps and compromise the user's privacy. This means that one of the journeys should be the default, if the user does not get referred by an advertisement and chooses not to take the test.

3.2.5 Requirements for the chosen personality feature

The main requirements for the architecture are:

- For every step of the workflow, it should be possible to define multiple variants of the contents of that step
  - It should at least be possible to change the description text of the step
  - It should also be possible to change other content of the step, such as images or other components of the webpage
- It should be possible to define multiple variants called journeys of the workflow page, which contain a variant for each step
- The selection logic should be able to respond to a request for the workflow page with the journey for the personality group the user is in. The user's personality group will be stored in a user profile in the database.

3.3 A/B Testing

This section is a report on the plan for the A/B test that we made during the project. The goals and requirements for the test, and the proposed plan are described in the following subsections.

3.3.1 Goals and requirements

The primary goal of the test was to evaluate the personalisation feature, so the client could make an informed decision whether to include this feature in the end-product. In order to do this, a significant increase in the evaluation metric of results would have to be measured between a variant with the personalisation feature and one without it.

From the first meeting, the client was enthusiastic about testing the interface and the personalisation functionality: it would be both an opportunity to test the viability of the personalisation feature, while at the same time receiving structured feedback from target users on other aspects of the interface, which became a secondary goal of the test.

Challenges in A/B testing in development If a product has no user-base, for example because it is in development and has not launched yet, A/B testing can still be done. There are some challenges and limitations to this. This was the case with Hypofund, so our own context is used as an example here:

First, test users have to be found to do the test. Finding users can be outsourced, but this can be expensive. The availability of test users is further limited if extra constraints are to be met, such as when a potential user has to be part of the target demographic of the product. In our case, the target customer is someone with some prior knowledge of mortgages, and ideally someone who is looking to buy their first house in one or two years. Another disadvantage is that test users who participate in these tests are in it for monetary compensation, which means that they have no vested interest in using the product or in the results of the product. In our case, the product does not give real mortgage advice, and test users are not looking for real advice in a test environment.

For the personalisation feature, the best metric would be the conversion rate, which is the percentage of visitors that close a mortgage. However, since the test users would be paid to fill in the forms from start to finish, measuring the conversion rate would give no indication of how the product would perform with real users. So, instead of using a conversion rate to find the value of the personalisation feature, we have to combine other metrics. Picking and combining these metrics introduces a major subjective element, since we decide what is important.

In meetings with the client, we came to the conclusion that the product is not in a stage that represents the end-product closely enough to gather useful feedback on it via a 200-300 users A/B test. To still validate the personalisation feature and the A/B testing architecture, we decided to do the test as a proof of concept with at least 20 test users.

3.3.2 Proposed test procedure

With the goals, challenges and requirements in mind, we designed and proposed to the client and coach the following specification for the A/B test. The test process is designed to measure the effects of variations in the customer journey (specifically the textual descriptions that are shown to the user) based on a personality test.

There are three main segments: the personality test, the customer journey and the evaluation survey.

Pre-test Via a link sent in an email or otherwise, the test user arrives on the index page of the website, which is hosted on a test server. Here, the user reads an introductory text, explaining the nature of the test. We mention that we aim to measure the impact of adapting content to personality traits, so we can justify the upcoming personality test to the user. We also mention that the user completing the test is more important than the accuracy of the information, and that they can also give estimates if they don't have certain information available (which is probably the case for some questions in the form). This also prevents pressuring the user to give out sensitive information, such as yearly income, alimony payments, etc..

In this introductory text, we explicitly state that the user agrees to the following terms of use if they start the test:

1. All information gathered in the text is private and will under no circumstance be given to third parties.

2. Any information we give in the test (such as the maximum mortgage calculations) may be different from reality

Once the user agrees and starts the test, they are randomly assigned to either test group or the control group.

Personality test In the first segment of the test, all users participate in a personality test, which determines the personality score. In the Hypofund end-product, a personality test will be included in the journey itself. As this would not be the case at the time of our test, we decided to measure the personality score before the user starts the journey. This effectively replaces the personality score measuring functionality in the journey.

As requested by the client, the personality test is based on the BSR personality model, a model developed by the Dutch SmartAgent marketing company. This model is similar to the HBDI model in psychology. [31] The reason for choosing this model is that it is directly aimed at grouping people based on the choices they tend to make, instead of on intrinsic personality properties. We only use the two opposing quadrants - yellow and blue - that are of interest to us in this model, since the other axis of the model is not of importance to the segmentation our customer wants to apply. The personality test should take five minutes.

The official test is based on five segments: in each segment the user is presented with a set of diverse values, jobs, hobbies, etc. and asked to select those that they identify with. Each option corresponds with one of four personality types, represented by colors.

The client, who had previous relations with the SmartAgent agency, asked a contact for the whole test specification, which they - understandably - could not give to us. This meant that we had use our own knowledge of the BSR model to copy the official test survey as well as we could. In our test, we only included those choices that - as far as we could tell - corresponded with yellow or blue. We iterated on our custom test with the client, which led to some changes, but overall he agreed with our choices. Since a major part of our client's work is understanding user choices, we think that his expert opinion provided enough guidance to construct a representative personality test.

After the test, the personality score is submitted and saved to the database, but remains hidden from the user. As stated, there are two customer journeys: a blue journey, which is tailored to users with a blue personality, and a yellow journey, tailored to yellow personality users. The only difference between the test and control variant is how the customer journey is assigned.

The users in the test group are each assigned the yellow or blue journey based on the outcome of the personality test: yellow test users are assigned the yellow journey and blue test users are assigned the blue journey. The users in the control group are assigned randomly, based on the ratio of yellow to blue test users.

Customer journey The test user participates in the customer journey. During this time, the following objective measurements were taken:

- The start and completion time of the journey
- The entered data in forms
- The submission timestamp for every step

After the final step of the journey is completed, the user is redirected to the last segment of the test, which is the evaluation survey.

Evaluation survey In the evaluation survey, we measured subjective aspects of the user experience that can be influenced by the personalisation by asking whether the user agrees with statements about the journey. Each statement the possible choices are a typical Likert scale with 'Strongly agree', 'Agree', 'Neither agree nor disagree', 'Disagree' and 'Strongly disagree'. The seven statements were:

1. The explanation and other texts were clear.
2. The explanation and other texts were too lengthy (Dutch: 'te langdradig').
3. It was always clear to me what information was being requested.
4. I have learned something new about mortgages.
5. The appearance of Hypofund appealed to me.
6. I would consider using Hypofund for mortgage advice (instead of hiring a mortgage adviser).
7. I would recommend Hypofund to others.

In line with the secondary goal, we also took the opportunity to add three more statements about specific interface functionality, and one open-ended question. These answers were not used for the results of the A/B test, but were there for the client's knowledge. We wrote the questions ourselves, but we iterated on the result with the client.

To find out whether the user was in the target Hypofund group, we ask whether they own a house, and if not, if they plan to buy a house within two years. By filtering the user group on only those who answered yes on either question, the results may become more relevant.

The last question asks whether the user knows somebody else who would like to participate, and to fill out their email address. After this, the results are submitted and the user is redirected to a 'Thank you' page. Here, a button can be clicked to start a new user test, to let subsequent people do the test on the same computer.

3.3.3 Assignment of journey in the control group

Users in the test group are assigned the journey that corresponds with their personality. Control group users are assigned a journey randomly, based on the ratio of yellow and blue population that our personality test finds. This way, the same amount of users did the yellow journey in the control group as in the test group (and the same goes for the blue journey).

Example The A/B test is done with 200 test users. The personality test finds that 150 of the 200 users are 'blue' (i.e. have a blue personality), and the remaining 50 are yellow: this is because the ratio of blue to yellow is apparently 75% in our test user base (and maybe even in the general population).

All the users are now evenly split in a test and control group, so each group now has approximately 75 blue and 25 yellow users. In the test group, the 75 blue users are assigned the blue journey, and the 25 yellow users are assigned the yellow journey. In the control group, the users are assigned journeys randomly based on the 75% ratio the personality test found: this way, approximately 75 users in the control group do the blue journey, and approximately 25 users do the yellow journey.

Dynamic ratio If the amount of test users is known beforehand, the ratio of yellow and blue users could have been found out by doing the control group first, at which point the ratio is not needed yet. The only disadvantage of this method is that would introduce bias in the results due to the "day of the week" effect. [18]

In our A/B test, the final amount of test users is unknown, so the blue and yellow ratio was initially set to a statistic we got from the creators of the BSR model. The problem with this ratio is that it may not stay representative for long in the target demographic of Hypofund. Also, the literature ratio is for the official BSR test, of which our test is only an adaption which might be different. So, after a certain amount of users the ratio should be tweaked, either by hand or programmatically.

3.3.4 Confounding variables

For the testing, the following list of interesting possible confounding variables was established:

- Personality profile (blue or yellow)
- Owns a house
- Plans to buy a house in the next two years
- Age
- Whether the user came from the circle of friends/friends family or not
- Computer knowledge
- Mortgage/financial knowledge
- Prior experience with mortgages (having closed one previously)
- Prior experience with mortgage advice
- Having used another online service previously
- Having used a physical mortgage adviser previously
- Sex

- Opinions on privacy
- Reading speed/Step completion speed
- Usage of profile page
- Browsing environment
- Browser
- Operating system
- Screen size/resolution

We collect data on the first three variables, and investigate the first one extensively (see subsection on data interpretation 3.3.6).

3.3.5 OEC

For our test, the collected metrics fall in two categories.

- The objective metrics, which are various completion times collected during the journey segment of the test
- The subjective metrics, which are the ratings of the first seven questions of the evaluation survey at the end

For the calculation of the OEC, we decided to only rely on the second category, the subjective metrics. This is because a fast completion time of the journey or separate steps could be interpreted as both a positive and a negative result. A fast completion time could indicate the user went through the process quickly without getting stuck, which leads to the better conversion rate. But a fast completion time could be interpreted as a negative result just as well: there was too much text, so the user did not read it, which leads to a worse conversion rate. If the completion time can be argued either way, we can't be sure (enough) whether it correlates with a higher conversion rate or not, so we decided to leave it out.

That leaves the seven evaluation questions, which are all of the same format. They are scored from 1 through 5, and a higher score means positive feedback (except in the second question, where a higher score is negative).

All questions carry the same weight, and all are required for the user to answer, so we made our OEC calculation very straightforward. We average the scores for every question, and then normalize the OEC to a value between 0 and 1. Our assumption is that an OEC closer to 1 corresponds to a higher conversion rate.

3.3.6 Data interpretation

Multiple analyses that can be performed on the collected data to come to conclusions and find patterns of interest. For drawing conclusions, the OEC is the primary factor. Afterwards the same OEC can be calculated for different groups based on possible confounding variables.

In the following interpretations, all the test users are simply categorized as either having a yellow or blue personality. In all these cases, more nuanced and

maybe more meaningful conclusions can be drawn if we account for the actual personality score (by assigning greater weights to those with higher personality scores) instead of the simple binary classifier of yellow and blue.

There are several reasons for this decision. First of all, there are also no gradations in the user interface, which is either blue or yellow (and nothing in between). Secondly, since the test was partly handmade by ourselves and the client, there is no evidence that the more precise score may have any meaning. Thirdly, because the time to do the data interpretation was limited, and a simple analysis between two groups take less time to implement.

A total of three conclusions were presented to the coach and client.

Conclusion 1: impact of personalisation This is the primary conclusion of the project. To support this conclusion, the mean OEC value for control group users and the mean OEC value for test group users can be compared. The significance of the results depends on the amount of users, but all the users count towards the significance (as is not the case with some of the other conclusions, where we can't include all test users in the calculations).

Conclusion 2a and 2b: impact of personalisation for the possible confounding variable yellow and blue personality A possible conclusion is that while there is no significant result by examining the mean OEC values for control and test groups, there still might be a significant difference in OEC for users with the one personality, but not for user with the other personality. For example, it might turn out that for users with a blue personality getting a personalised journey does not significantly affect their OEC, but for users with a yellow personality a significant increase in OEC is measured when they get a personalised journey (compared to when they get a random journey).

To support this conclusion, we can compare the OEC value of the test group users that have a yellow personality (and who have consequently done the yellow journey) and compare them to the mean OEC value of the control group user that have a yellow personality. In this calculation, the results will have to be more pronounced to be significant, because the results of users with a blue personality are not applicable.

The same calculations can be done for the possible confounder 'blue personality'. This would be conclusion 2b.

Conclusion 3: impact of journey This conclusion is different from the previous ones in that it is not intended to measure the impact of personalisation on the customer experience. Instead, the two journeys are compared without taking user personality into account, to find out if in general a blue journey is better than a yellow journey. If in the future the client does not want to use the personalisation functionality, he can still make an informed decision of which journey to use should he include only one.

Which journey is best can be measured by comparing the OEC of the control group users that have done the yellow journey with OEC of the control group users that have done the blue journey.

So, to support this conclusion, only the results of the control group can be used, which is only half of the data. However, if the results for conclusion 1, 2a or 2b turn out to be insignificant, it can be concluded that personalisation has

no effect at all for users with either or both personality type. In those cases, that data can also be used - in addition to the control group data - to increase the significance of conclusion 3.

4 Implementation report

In this section, we report on how the interface, personalisation and A/B test was implemented and realised, on a sprint by sprint basis. In total, there were 3 sprints of 1 week each, and a final sprint of 2 weeks.

4.1 General implementation requirements

The client had the technical requirements that the project should be developed in the Python 2 programming language, with Django as the core web-development framework. This is because it should be possible in the future to integrate our code with the core Hypofund codebase.

Django is a web-development framework with an MVC like architecture. Django's architecture consists of three main parts: models, views and templates. The model classes specify the tables of the database, in our case PostgreSQL, and the type, restrictions and other options of the fields. These fields are mostly accessed by the view code to save and retrieve data. The view code are classes or functions that handle the HTTP requests and render Django template files. Templates are like HTML, but can also use logic and variables. Once rendered, they are sent to the client's browser.

4.1.1 Library choices

Libraries and web development tools are chosen with care, as each has to be compatible with the Hypofund code base, be able to fulfill the requirements and should integrate well with Django. Bootstrap [1] was used because of its ease of use and the required interface had a minimal design. For supporting icons, Font Awesome [7] was used. The CSS enhancing library SASS [8] was used. To render the forms, Crispy Forms [3] was advised by the tech lead of Hypofund, since it integrates well with Django.

4.1.2 Testing

For writing tests, we used a combination of unit-tests and end-to-end tests. Compared to the usual test pyramid [15] we wrote a large amount of end-to-end tests using Selenium [10], and a low amount of unit tests. The reason for this is that much of our personalisation leverages Django's templating system, meaning there is relatively little logic to test, but many changes in the interface. Combined with the client-side javascript used to dynamically load new steps through AJAX calls, this means our interface is bug-prone, while our server-side logic is relatively robust. Increasing the amount of end-to-end tests compared to unit tests helps focus our testing on parts of the system that are the most susceptible to bugs.

To determine what tests to write, we used a combination of requirement-based testing, functional testing and line coverage analysis. Our end-to-end tests were functional in nature, since they have no knowledge of the internals of our system, but only tests the resulting interface. We used requirement-based testing to determine what functionality was most important, and wrote end-to-end tests for this functionality. Finally, for the unit tests line coverage analysis was used to make sure that all our essential logic was tested.

4.1.3 Other choices

For version control Git was used, supported with GitLab. We used Scrum during the entire process, with a digital Scrum board on Trello and the Trello Scrum extension. During the first two sprints we also used a physical Scrum board in the office, but updating two boards turned out to be too impractical.

At the start of the project, our code interfaced with the Content Management System django-fluent [5] written by the lead programmer for Hypofund. After the first sprint, it was decided to remove the integration with fluent, since it was not needed for our prototype and added extra complexity.

4.2 Sprint 1

In preparation for the first sprint, we started working on multiple Python and Django tutorials, and read the documentation provided by the client. In the beginning of the sprint we spent some time getting to know the infrastructure that had been set up by the client. The learning curve of Django was steepened, because there was also the CMS with which we had to integrate.

We started development by determining what steps the user journey consists of, and what functionality they should contain.

4.2.1 Steps

Each part of the journey that consists of a related set of information that the user has to fill out, is called a step. There are 9 steps in our version of Hypofund:

1. Personal data: essential personal data needed for a mortgage advice, such as birth date
2. Risk profile: 5 questions that are used to determine the user's attitude towards financial risks
3. Maximum mortgage: financial data, such as yearly income, needed to calculate the maximum mortgage the user can get
4. Mortgage sum: information about the house the user would like to buy, used to determine what the total mortgage sum for that house is
5. Mortgage type: choice of the type of mortgage, fixed-rate or linear
6. Fixed interest rate period: choice between 8 possible options for interest rate fixation period, allowing the user to limit risk due to changing interest rates
7. Term life insurance: allow user to choose 2 possible options of term life insurance
8. Living cost insurance: allow user to choose 3 possible options for living cost insurance
9. Compare offers

Each of these steps may contain the following parts:

1. The title of this step
2. Introduction text, explaining what the purpose of this step is, what type of information the user has to enter, and what type of choice the user has to make
3. A form allowing the user to enter the required data and make the required choices
4. Description text for possible choices a user can make
5. The result we have calculated from the data the user has provided

A step does not necessarily need to have all of these parts, it can have just one or several of the parts. However, the parts are always shown in the same order.

Because of the steep learning curve of Django, we implemented the easier steps of the workflow first, which were the fixation period step, the mortgage type step, the term life insurance step and the living cost insurance step.

4.2.2 Asynchronous loading of steps

Once we had built some steps, we implemented the architecture for displaying the steps one by one, which is an important part of the hybrid interface design.

We used server-side rendering, in order to leverage the django-templating system and keep client-side logic to a minimum. After the client posts the data of a step to the server, the server performs validation on the data, and if the data validates saves it to the database. The server then returns two blobs of HTML to the client: HTML for the submitted step, and if the data validated also HTML for the next step. The HTML for the submitted step is needed to show any validation errors, remove previous validation errors, and update the entered data to a normalized format.

After the client receives the server response, it checks the HTTP status code: 200 success or 400 bad request. If the server responded with success, the client updates the HTML of the submitted step and loads the HTML of the next step. The page then shows the result panel of the submitted step, or the first panel of the next step (if the submitted step does not have a result). If the server responded with bad request, the client only updates the HTML of the submitted step to show validation errors.

4.2.3 Landingpage

We worked on the landing page and the mortgage calculator3.1.3 element for the first two sprints, before the feature was dropped. We decided to cut the landingpage for the benefit of the user test, because we expected that the landingpage would only distract from the workflow. We created the front-end for the landing page and the static forms for the mortgage calculator.

4.2.4 Sprint retrospective

Estimated points: 17
Consumed points: 27

During this sprint, a lot of time was invested in design meetings and learning the existing codebase and Django. We did not plan those activities as part of the sprint, since they were hard to plan in advance, which explains our low points estimate for this sprint. This was partly accounted for by making a low estimating the story points, but we still consumed 10 more points than estimated. For next sprint, we would have to make substantial higher estimates.

The client had the comment that the demo should be better prepared, as something went wrong right before the demo. We also decided to have a demo online on a test server, so that the latest release was always available for the client. Another point of improvement was to write better user stories on the online Scrum board, since our descriptions were sometimes unclear to the client.

4.3 Sprint 2

4.3.1 Technical debt

In the first sprint, we built up some technical debt, which we had to handle in the second sprint. This included refactoring the server-side code, since we had to be ready to implement the personalisation architecture. The client-side code also caused bugs and was not very maintainable, so we spend time refactoring that code too.

4.3.2 Personalisation

The primary feature that was implemented was that the first iteration on the personalisation, that is, the functionality to switch between journeys3.2.4.

Every part of a step may have multiple variants, in the version we build there will usually be two. Any part should also be able to have just one variant, so the variants for different parts of the step should not be coupled, to allow each part to have a different number of variants.

The specification of the variants should be manageable and easy to update. To achieve this, which variants and parts a step has is decoupled from the content of the variants. We achieved this by specifying separate dictionaries, one specifies what parts and part-variants a step has, and the other specifies where the content for a specific step-part-variant can be found. We later moved this configuration to a separate data file, in order to prevent mixing configuration and logic.

Because the data structure was well specified, implementing the logic to get the correct data for a variant was relatively easy. It was harder to integrate the variants with the existing code to load new steps through AJAX calls from the client, and this code became overly complex and somewhat buggy.

4.3.3 Workflow roadmap and profile

The roadmap and profile elements were improved. The workflow roadmap was implemented, which displays on which step the user currently is. The roadmap and profile elements were fixed on the screen, so that they scroll down with the user. We also iterated and agreed on the profile design.

4.3.4 Bugs

Along with other bugs, two major bugs were caused by the workflow being dynamic instead of static. The client side event handlers had to be reattached, every time the workflow was updated, and there was a problem with the form validation which caused validation errors not to be shown under some circumstances.

4.3.5 A/B testing

In preparation of A/B testing, we compared options for the implementation that were compatible with Django, such as django-experiments [4], django-lean [6] and django-ab [2]. The first one had the features we needed, and had recent updates, unlike the other two. However, the only metric that it could effectively detect was the conversion rate, which was not the metric we were after. This led to our decision to implement the A/B testing architecture ourselves.

4.3.6 Sprint retrospective

Estimated points: 64
Consumed points: 64

By doubling our initial estimates, at the end of sprint 2 we were spot on. This was also thanks due to our increased velocity, and scaling the learning curve of Django.

4.4 Sprint 3

4.4.1 Technical debt and bugs

We started sprint 3 by refactoring some of the code we thought could be structured better from sprint 2. This took a significant amount of time, but was very successful in preparing for new features and getting rid of bugs. We converted our server-side view code to use Django class-based views, at the same time removing unneeded complexity by separating duplicated logic into separate functions. We also overhauled our client side javascript to separate low-level functions doing DOM manipulation from high-level functions which respond to the server requests. This got rid of all our major client side bugs.

The deadline for the first SIG submission was at the end of sprint 3, so we also spent a substantial amount of time on refactoring based on the SIG guidelines such as avoiding code duplication and modularisation.

4.4.2 Workflow steps

We implemented the rest of the workflow steps. The risk profile step was partly implemented by the technical lead of Hypofund, because this required advanced knowledge of the CMS and Django forms. We implemented a visual representation for the risk scores, so that a score was represented as an amount of points instead of a number.

The variant specification functionality was extended to be able to specify a Django template to include for each variant, instead of just specifying a string.

This functionality was essential to be able to show the results of calculations in the result parts.

The step for comparing mortgage offers was extra work, because it had to convincingly fake a calculation of finding the offers with delays and spinners.

4.4.3 TypeForm

For the personality test and the test survey we used a survey tool called Typeform [11]. Even though some workarounds were needed to get all the functionality we needed, this was faster than implementing the whole form survey with Django.

4.5 Sprint 4

Sprint 4, which spanned two weeks, was about getting everything ready for the user test and performing it. Most of the content was written (in collaboration with the client), the profile was fully implemented and many hours were devoted to fixing all known bugs, which were for the most part hidden in the interface code. In multiple demo's, the client gave feedback, which was incorporated in the final interface.

This sprint spanned two weeks, since we decided to do the proof-of-concept user test on the Thursday of the first week of this sprint. This caused us to work on many tasks on this sprint that could not be planned in advance, which is why we decided against doing an extra sprint planning.

4.5.1 Profile and roadmap

The functionality for the profile was completed. The profile summary could be expanded to the full profile, that displayed the inputted data of the user in collapsable panels for clarity. The panels that have no information to show yet cannot be expanded. The transition from profile summary to full profile and back was animated.

4.5.2 Selenium end-to-end test

On the recommendation of the TU Coach, we also wrote Selenium end-to-end tests in this sprint, which verify that essential functionality works correctly for all variants. There were some problems with getting this to work with GitLab's continuous integration, but overall they were an useful addition because of the dynamic nature of the product.

4.5.3 Bugs

A lot of time was spent fixing all known bugs, since any bug would be detrimental to the user test. The A/B testing and the personalisation feature worked without problems, but there were many time consuming bugs with interface.

4.5.4 A/B test pages and redirects

Before the workflow, we added a page with an introductory text and two conditions for participating that the user had to agree with. We created the personality test and the evaluation surveys with Typeform, which we added to the

proper pages before and after the workflow, respectively. The personality test and evaluation survey was written in collaboration with the client.

To make sure the test users were always on the page we wanted them to be, we made sure to create redirects: for example, an user that already did the personality test that request the personality test URL (for example, by using the browser's back button on the workflow), gets redirected back to the workflow.

4.5.5 Small interface changes

We polished the interface by making small changes and additions:

- The site is for the most part responsive
- The user is able navigate by clicking on completed steps in the roadmap
- When the user submits information or clicks on 'next' button in the workflow, the screen scrolls to the next panel
- When the user returns to or refreshes the workflow, the window scrolls to the last step
- Some forms are hidden until a specific checkbox is clicked
  - Example: only when the checkbox 'Do you have student debt?' is checked, the input form 'How much student debt do you owe?' appears

4.6 Sprint 4, second week

Because the logic for the user test meant that the journey for a user was now stored in the database, we changed the switching logic to use this value from the database, instead of the URL.

In the second week of sprint 4, we wrote the final content for both journeys, in close collaboration with our client. We also integrated the variant code further with the user test code and carefully tested that the variant logic was correct and consistent under all possible test conditions.

5 Evaluation

In order to evaluate both the interface we built as well as the personalisation feature, we ran an A/B user test, based on the plan as explained in section 3.3. In order to run the A/B test, we used the tooling we built, as described in the A/B testing subsections of Sprint 2, Sprint 3 and Sprint 4 in the Implementation report. The following sections describe how the test was set up, how we ran the test, and how the results were analysed. Finally, we report the results of the evaluation.

5.1 Test set-up

After the main implementation, we determined together with the client that our prototype was not in a state that was representative enough for the final product to run a full-scale user test. Running the test with the planned 200-300 users would require a significant investment from our client, because he would have to buy a user panel with representative users. Together with our TU coach and client we decided to do a small scale test with friends and family instead.

5.1.1 Goal of small-scale test

The small-scale test functions as a proof-of-concept that the testing setup works. As a secondary goal, there is a chance that we can draw conclusions from the test about the effect of the personalisation feature, even with a smaller sample size. We predict that the sample size will be too low to draw a significant conclusion about our main test hypothesis, however it is still possible that there is a significant result when controlling for confounding variables, or when looking at the results of individual evaluation questions.

We determined together with our TU coach that in order to be able to apply the planned statistical tests, we would need at least 20 users to test our main hypothesis (10 in both the test and control groups). However, to test for possible confounding variables, we would need to divide the entire test population in 4 groups (2 groups split on the confounding variable for both the test and control groups), bringing the minimum amount of test users to 40.

5.2 Running the test

5.2.1 Finding users

The test users were found in two ways. We used opportunity sampling to find users in our own circle of friends and family. To get a more representative sample of the general population we were planning to ask people from the office building of the client, which were closer to the target demographic of Hypofund.

However, this turned out to be impractical, so we organised a user test in a coffee place in the city center of Utrecht instead. We offered coffee to people on the street in exchange for their participation in the test. This new plan was a success, because on a € 42,50 budget (provided by the client) we found 18 users in 3.5 hours time.

5.2.2 Feedback during the test

Since we started out the test with users we found in the center of Utrecht, we were able to verify that they were able to make the test successfully, without running into significant issues that would affect their ability to evaluate our prototype properly. Apart from minor practical issues, like the laptops we were using locking the screen after a period of inactivity, there were no reported issues.

Once we verified that the test was working as intended locally, we sent invitation links to friends and family using WhatsApp, Facebook, and personal invitations. This resulted in a total of 29 responses over three days time from the afternoon of Friday the 10th of June until the afternoon of Monday the 13th of June, bringing the total amount of responses to 47.

Two users were unable to make the test, due to an issue where they were unable to submit the first step on the workflow page. Both of these users were on company networks, so we suspect that a company firewall or other type of network filter might have affected the test. One of the two users was able to complete the test later at home.

5.2.3 Dynamic blue yellow ratio

Even though a system was proposed that tweaked the ratio after every user, it was never implemented due to time constraints. We ended up fine-tuning the ratio during the test ourselves. For the first 29 responses, the ratio was set at 50/50 due to an unpushed update. Then we updated the ratio to 40% blue, 60% yellow, which was based on a statistic we got from SmartAgent. After 10 more responses we tweaked the ratio to 55% blue, 45% yellow to get as close as possible to the ratio of the personality test. This approach was moderately successful, since the ratio of blue/yellow journey for the control group was 10 blue/13 yellow for an overall bsr profile ratio of 25 blue/22 yellow.

5.3 Analysis

All analysis was performed according to the principles outlined in the research section on A/B testing (2.3), in particular using the statistical tests outlined in the Statistical analysis of OEC section (2.3.2). We used SciPy [9] for all statistical analyses. We started out calculating the OEC as described in the Agreement on what to build > A/B testing > OEC section for the groups relating to the three main hypotheses we wanted to test, as outlined in the Agreement on what to build > A/B testing > Data interpretation section. After reviewing the data, we suspected that there may be another confounding variable: being in the strangers or family and friends group. The strangers group consists of people we offered a cup of coffee in the center of Utrecht to make the test, while the friends and family group are people we found using opportunity sampling amongst family and friends. We decided to add an extra test to check whether this variable is a confounder, as specified in conclusion 4.

5.4 Results and Conclusions

Combining the group strangers (n = 18) of users we found in the center of Utrecht and friends and family (n = 29), there were a total of n = 47 users in

our test. These users were all randomly assigned to either the control (n = 23) or the test (n = 24) group.

5.4.1 Conclusion 1: impact of personalisation

- H_0: test and control groups do not yield different OEC scores
- H_1: test and control group yield different OEC scores

The chance for the null hypothesis to be true (p-value) using the Mann-Whitney-Wilcoxon test is 0.1090 with a test statistic of $U = 201.0$. For our confidence level of 95%, this means we have failed to reject the null hypothesis.

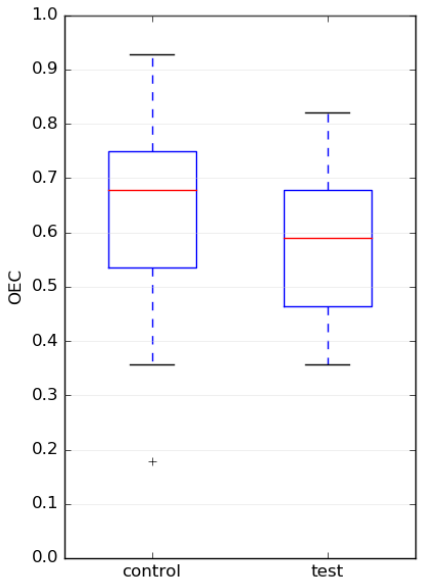

Figure 1: Boxplot of OEC for control versus test group

5.4.2 Conclusion 2: impact of personalisation for the possible confounder yellow versus blue personality

- $H_{0,\text{yellow}}$: test and control groups for yellow personalities do not yield different OEC scores
- $H_{1,\text{yellow}}$: test and control groups for yellow personalities yield different OEC scores
- $H_{0,\text{blue}}$: test and control groups for blue personalities do not yield different OEC scores
- $H_{1,\text{blue}}$: test and control groups for blue personalities yield different OEC scores

The chance for both null hypotheses to be true (p-value) using the Kruskal–Wallis test is 0.3228 with a test statistic of $H = 3.485$. This means we have failed to reject both $H_{0,\text{yellow}}$ and $H_{0,\text{blue}}$.

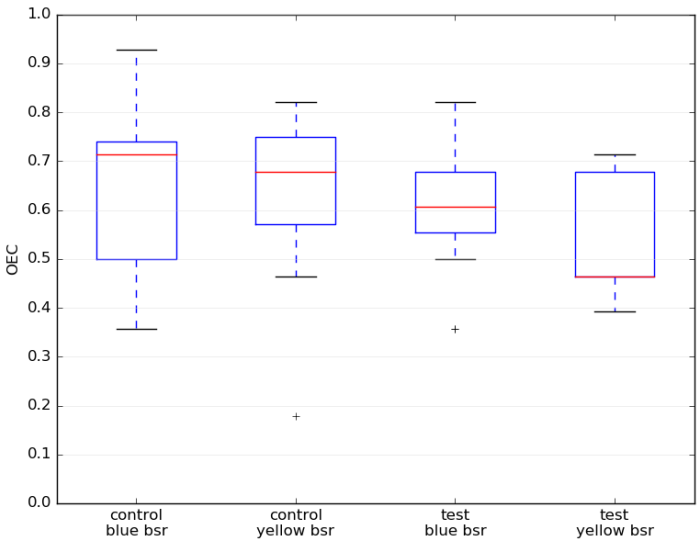

Figure 2: Boxplot of OEC for control versus test group with BSR profile as possible confounding variable

5.4.3 Conclusion 3: impact of journey

- H_0: control yellow journey and control blue journey groups do not yield different OEC scores
- H_1: control yellow journey and control blue journey groups yield different OEC scores

The chance for the null hypothesis to be true (p-value) using the Mann-Whitney-Wilcoxon test is 0.3660 with a test statistic of $U = 79.5$. This means we have failed to reject the null hypothesis.

5.4.4 Conclusion 4: impact of personalisation for the possible confounder strangers versus family and friends

- $H_{0,\text{strangers}}$: test and control groups for strangers do not yield different OEC scores
- $H_{1,\text{strangers}}$: test and control groups for strangers yield different OEC scores
- $H_{0,\text{family and friends}}$: test and control groups for family and friends do not yield different OEC scores
- $H_{1,\text{family and friends}}$: test and control groups for family and friends yield different OEC scores

The chance for both null hypotheses to be true (p-value) using the Kruskal–Wallis test is 0.3726 with a test statistic of $H = 3.126$. This means we have failed to reject both $H_{0,\text{strangers}}$ and $H_{0,\text{family and friends}}$.

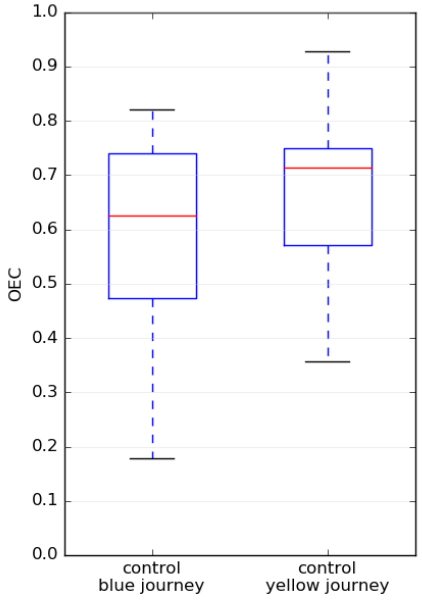

Figure 3: Boxplot of OEC for control blue journey versus control yellow journey groups

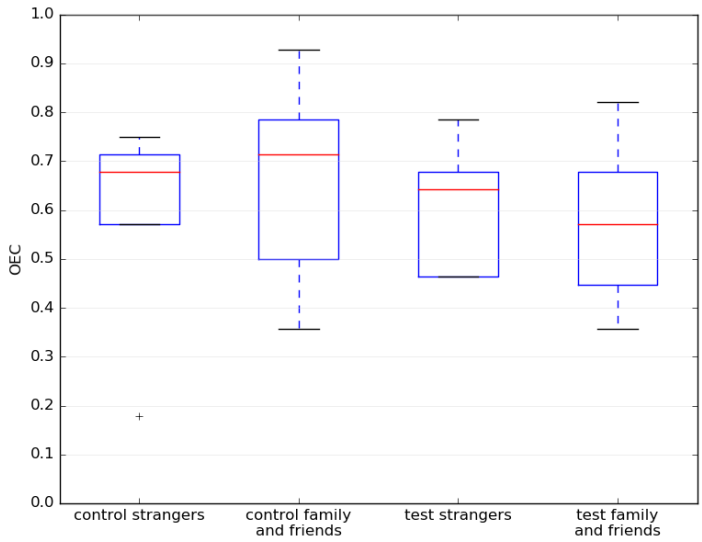

Figure 4: Boxplot of OEC for control versus test group with strangers versus family and friends as possible confounding variable

5.4.5 Overall conclusion

As we had expected beforehand, none of the results we found were statistically significant. The result that was closest to significance was for conclusion 1: the OEC for control versus test group, with a p value of 0.1090. However, the

median OEC for the test group is actually lower than the median OEC for the control group, opposite to our expectation. Possible explanations for this will be given in the discussion.

6 Discussion

The results of the A/B test are non-conclusive: there is no significant indication either way that the personalisation feature as implemented does or does not improve the OEC. However, even with the small sample size, it seems unlikely that performing a large scale test will give different results, because there is not even a small difference to be found, and there are some uncertain elements in the test which we will discuss in this section. These reasons lead us to advise the client against investing in a larger A/B test at this time.

Instead, we recommend that the A/B test and personalisation feature be improved first, as there are a number of uncertain elements. These elements should either be removed, or singled out and tested separately to ensure their effectiveness. Once these issues are fixed, we think a large scale test has a good chance of producing significant results.

The first uncertain element is the BSR personality test. This test was based on an existing test, which was adapted by ourself and the client. As mentioned in section 3.3.2, we never got hold of the actual specification. This introduces a level of uncertainty in comparison with the original test. Furthermore, although the BSR test has been effective in marketing, we could not find an error rate for the test in scientific literature.

The second element was the content, of which both variants were written by ourselves (and reviewed by the client). We are of course not professional writers (or marketeers), and our knowledge of the BSR personality types is limited. In addition, the changes in variants could have been implemented in more than textual change alone, such as for instance via images, colors, etc.. As it is now, the difference between variants is rather small, which decreases the chance of significant results.

The third and final uncertain element was that the OEC was based on a subjective evaluation survey, which tried to predict conversion rate. This evaluation was written by ourselves (and reviewed by the client), and it is not tested whether a positive score on the evaluation actually corresponds with a positive experience. It might very well be that we are asking the wrong questions, or that some questions are more important than others and should be weighed accordingly. To take this last uncertainty out of the equation, the A/B test could be performed in production when the conversation rate can be measured directly. After all, an important goal of an A/B test is to measure what works, instead of relying on human predictions of what works.

Once these issues are fixed, it would be wise to perform another small scale (and low investment) proof-of-concept test run, since we have experienced that this is a good way to confirm or deny the validity of the test. In addition, this makes sure that implementation-wise the whole system is ready for a large scale test.

6.1 Discussion of primary conclusion

As mentioned, the results for the primary conclusion were the strongest. Even though this result is not significant, we decided to investigate it further, since its implications could be important for further research.

We determined the following hypotheses that could cause this result:

1. People reacted negatively to getting a personalised journey
2. People we classified as yellow are actually blue and vice-versa
3. Yellow people actually like the blue journey better and vice-versa

Hypothesis 1 seems impossible, since test users did not know whether they were getting a personalised journey. We did not have any indication during the test, or from the test results that people knew whether they were getting a personalised journey or not. Hypothesis 2 is possible, since the personality test is an uncertain element, as discussed in the previous section. However, although it is possible that our test was not very powerful, but we do not think the test could have consistently misclassified people. Hypothesis 3 is possible, and an interesting option to keep in mind. We have no way of verifying whether this might be the case with our current test, except that there is no statistically significant result that indicates it is.

This results highlights the importance of being sure that the content used for personalising actually fits the profile you are personalising to. Once a personalisation feature is implemented, it may be attractive to use it whenever possible, and write as much personalised content for this. However, without investigating whether the personalised content actually has a positive effect on user experience, the opposite may be true, where the personalised content actually has a negative effect.

6.2 Current state of Affective Interface research

Many research studies show interesting results for affective interfaces. Also, the need for this research has been competently argued for by pioneers of the field such as Picard. However, we have noticed that there are roadblocks that hinder the application of the techniques discussed.

The biggest roadblock is that, currently, and for so far we could find, research is hard to apply in practical, real-world products (such as Hypofund). This is primarily because the research cases tend to make use of specialized hardware (e.g., pressure-sensitive mice) or not readily available data (e.g., voice or video data). Maybe in the future, users are used to talking to their computers, or pressure-sensitive mice will have become mainstream and the researched techniques can be put to use.

We would like to see research that focuses on developing affective interface techniques that could be incorporated in present day products. We think the best way to do so is by developing techniques that only use the capabilities of the modern web browser. Making progress on such techniques, and using them in real-world products may boost interest in affective interfaces, which in turn could lead to more research. Also, tests of the new techniques when employed in accessible, online products could be done more easily, on a larger scale, and outside the laboratory setting.

For instance, one option we have not seen in current research was measuring user impatience based on the user skipping texts. Take, for example, an interface where blocks of text are served to the user one by one, the next text block appearing each time the user clicks a 'next' button after finished reading the previous block - as in the case of Hypofund. A system could be designed that would measure the user's reading speed, based on the amount of words per

block (or text complexity). If a user clicks the 'next' button faster than they could have possibly read the text with their measured reading speed, this could be interpreted as a sign of impatience. This system would be an example of an affective interface, which only makes use of technologies available to any modern web browser.

6.3 Current state of Adaptive interface research

The research on adaptive interface design is more applicable to real-world systems and formalized, with systemic approaches of how to design adaptive interfaces. Affective interfaces have not reached this level of concreteness yet, probably because adaptive interfaces have been in widespread use (such as the Facebook timeline, which prioritizes stories on relevance for the user) on the internet.

Affective and adaptive interfaces seem a natural fit: interfaces adapting to the user's affect could have great potential. However, from the papers we have read we have gotten the impression the research from both areas seem largely separate. The same can be said for context-aware interfaces, of which we saw no mention in affective or adaptive interface design papers. We think there are possibilities to develop new techniques and gain new insights if these fields would cooperate more closely in their research.

6.4 Reflection on our process

The process was overall smooth, although there were times at which we had to be flexible and realistic of the direction of the project. For example, when the test was scaled down, we had to adjust our expectations and the product to still be able to draw interesting conclusions.

At the end of the project, we had an evaluation and reflection meeting with the client. The client described the whole process as a 'learning experience'. From both sides the technical complexity of the project was underestimated. As discussed, for us this was particularly true with learning the Django framework, and for the client with the complexity of a software project.

The client also had feedback on asking and answering questions. He told us that when answering questions, we had the tendency to come up with an answer instantly. Instead, it is important to get to the heart of why we are asked this question in the first place: understanding the 'question behind the question'.

Conversely, when we were asking questions ourselves, it was sometimes important to ask more open ended question, instead of attaching multiple choice answers to each question. This forces us to take a more vulnerable position, which can be good for the process. This promotes creative discussions, which leads to answers we have not considered yet ourselves.

7 Conclusion

This project consisted of three main parts. First, we designed, proposed, and built a prototype for the interface of the Hypofund online mortgage advice platform. Then, we built a content personalisation feature, which adapts the content of the interface to the user's personality profile. Finally, we evaluated the personalisation feature we built with an A/B user test. This A/B test only ran at a small scale due to budgetary constraints, which is why it was only a proof-of-concept and why we did not expect significant results from it.

The design of the interface was successful, as our client was happy with the 'hybrid workflow' interface we proposed. This interface combines the advantage of a clear set of steps for the user to follow with the flexibility, easy overview and recurring value of a profile. The implementation of the interface went well, although it took more time than expected. The main reason for this was an underestimation of the complexity and the new technologies we had to learn. Even though the design of this interface was basic, it served its purpose well: to test the personalisation feature.

Implementing the personalisation feature was mostly successful. Even though time was very limited, we managed to implement a working prototype for the core personalisation features. The personalisation adapts the content shown in the interface to the personality profile of the user, as measured by a personality test.

Finally, the proof-of-concept A/B test yielded no significant results. This can mean one of two things: either the test did not have enough power to show a difference in user experience due to personalisation, or the personalisation feature did not work. We discussed several factors due to which our personalisation might not have worked, mostly related to uncertainties in the personality test we used and the content we used for personalisation.

Even though it is disappointing not to get more results, this is what we expected for a small-scale experiment like this, which is why it was only intended as a proof-of-concept. It served its purpose well, and the test ran successfully. Running the test showed us how important it is to evaluate all properties of a feature properly, especially when its effects may be uncertain. We showed that it is possible to run an A/B test using the methodology we used, and discussed how this methodology may be improved to yield more results in a larger scale A/B test

8 Bibliography

References

[1] Bootstrap. https://web.archive.org/web/20160622202007/https://getbootstrap.com/. Computer software.

[2] django-ab. https://web.archive.org/web/20160622201122/https://github.com/johnboxall/django-ab. Computer software.

[3] django-crispy-forms. https://web.archive.org/web/20160622202319/https://github.com/maraujop/django-crispy-forms. Computer software.

[4] django-experiments. https://web.archive.org/web/20160622201110/https://github.com/mixcloud/django-experiments. Computer software.

[5] django-fluent. https://web.archive.org/web/20160622200911/https://django-fluent.org/. Computer software.

[6] django-lean. https://web.archive.org/web/20160622201115/https://github.com/e-loue/django-lean. Computer software.

[7] Font awesome. https://web.archive.org/web/20160622202228/http://fontawesome.io/. Computer software.

[8] Sass. https://web.archive.org/web/20160622202300/http://sass-lang.com/. Computer software.

[9] Scipy. https://web.archive.org/web/20160622201815/https://www.scipy.org/. Computer software.

[10] Selenium. https://web.archive.org/web/20160622200856/http://www.seleniumhq.org/. Computer software.

[11] Typeform. https://web.archive.org/web/20160622201313/https://www.typeform.com/. Web service.

[12] Henk Bierlee, Olivier Dikken, Kilian Grashoff, Joe Harrison, Stephan Dumasy, Tim Molenaar, and Jonathan Raes. On the state of mobile interruptibility-aware systems. Unpublished manuscript, 2016.

[13] Frederik Michel Dekking. *A Modern Introduction to Probability and Statistics: Understanding why and how*. Springer Science & Business Media, 2005.

[14] Florian N Egger. Affective design of e-commerce user interfaces: How to maximise perceived trustworthiness. In *Proc. Intl. Conf. Affective Human Factors Design*, pages 317–324, 2001.

[15] M Fowler. Test pyramid. https://web.archive.org/web/20160622200549/http://martinfowler.com/bliki/TestPyramid.html, 2012. Accessed: 2016-06-22.

[16] Erich Gamma. *Design patterns: elements of reusable object-oriented software*. Pearson Education India, 1995.

[17] Michael Knappmeyer, Saad Liaquat Kiani, Eike Steffen Reetz, Nigel Baker, and Ralf Tonjes. Survey of context provisioning middleware. *IEEE Communications Surveys & Tutorials*, 15(3):1492–1519, 2013.

[18] Ron Kohavi, Randal M Henne, and Dan Sommerfield. Practical guide to controlled experiments on the web: listen to your customers not to the hippo. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 959–967. ACM, 2007.

[19] Ron Kohavi and Roger Longbotham. Online controlled experiments and a/b tests. *Encyclopedia of Machine Learning and Data Mining, C. Sammut and G. Webb, Eds*, 2015.

[20] Ron Kohavi and Matt Round. Front line internet analytics at amazon. com, 2004. *Presentation at the Emetrics Summit*, 2004.

[21] Derya Öztuna, Atilla Halil Elhan, and Ersöz Tüccar. Investigation of four different normality tests in terms of type 1 error rate and power under different distributions. *Turkish Journal of Medical Sciences*, 36(3):171–176, 2006.

[22] Nikolaos Partarakis, Constantina Doulgeraki, Asterios Leonidis, Margherita Antona, and Constantine Stephanidis. User interface adaptation of web-based services on the semantic web. In *International Conference on Universal Access in Human-Computer Interaction*, pages 711–719. Springer, 2009.

[23] Rosalind Wright Picard. Affective computing. 1995.

[24] Thomas S Polzin and Alexander Waibel. Emotion-sensitive human-computer interfaces. In *ISCA Tutorial and Research Workshop (ITRW) on Speech and Emotion*, 2000.

[25] Nornadiah Mohd Razali, Yap Bee Wah, et al. Power comparisons of shapiro-wilk, kolmogorov-smirnov, lilliefors and anderson-darling tests. *Journal of statistical modeling and analytics*, 2(1):21–33, 2011.

[26] Carson Reynolds and Rosalind W Picard. Designing for affective interactions. In *Proceedings from the 9th International Conference on Human-Computer Interaction*, 2001.

[27] Eric Ries. *The lean startup: How today's entrepreneurs use continuous innovation to create radically successful businesses*. Crown Books, 2011.

[28] Gustavo Rossi, Daniel Schwabe, and Robson Guimarães. Designing personalized web applications. In *Proceedings of the 10th international conference on World Wide Web*, pages 275–284. ACM, 2001.

[29] Kristina Schaaff, Raphael Degen, Nico Adler, and Marc TP Adam. Measuring affect using a standard mouse device. *Biomedical Engineering/Biomedizinische Technik*, 57(SI-1 Track-N):761–764, 2012.

[30] Matthias Schneider-Hufschmidt, Uwe Malinowski, and Thomas Kuhme. *Adaptive user interfaces: Principles and practice.* Elsevier Science Inc., 1993.

[31] Matthijs Wolters, Douwe Reitsma, Adriaan Lamme, Brand Hop, and Ernst Jan Reitsma. Hbdi vs. bsr: Een kritische vergelijking van twee segmentatiemodellen. *Human Systems Management*, 2:329–331, 2007.

Appendices

A Requirements

This appendix contains the list of requirement that were proposed to the client. The functional requirements are divided in the four MoSCoW categories: Must Have, Should Have, Could Have and Won't Have. The requirements have been derived from the hybrid user interface, which is in line with the Hypofund goals and requirements as have been outlined in the Goals and Requirements section.

A.1 Functional requirements

A.1.1 Must have

- Landingpage
  - The user should see clear branding of Hypofund (logo, slogan, etc..)
  - The user should be able to enter information in simple forms on the landingpage, such as yearly income and building price
    - This form should be positioned above the fold
  - The user should be able to click a call-to-action link that leads to the workflow
  - Under the forms there is space for information about the company and products of Hypofund
- Workflow page
  - The page should contain the profile summary, the workflow steps and the roadmap
  - Profile
    - The user should see a static profile-summary, with the most important information he has entered thus far
    - The user should see a standard avatar profile picture in the profile summary
  - Workflow steps
    - All steps are implemented with a simple interface and are functional
    - The user can input information in a few simple steps
      - Text-only steps
      - Steps with (toggle-able) forms
    - When the user completes a step, the next step appears
  - Roadmap
    - The user should see the roadmap, which gives the user a high level overview of the workflow steps
  - A user should never have to input the same information twice

A.1.2 Should have

- Landingpage
  - The forms on the landingpage should calculate the maximum mortgage and return this information to the user
  - The forms on the landingpage should calculate whether the house with the given price can be bought with the maximum mortgage
- Workflow page
  - The profile summary grows dynamically whenever the user completes a step
  - The profile summary has a button which expands the summary to the full profile
  - The user should be able to change his profile avatar
  - The roadmap should indicate at which step the user is, by dynamically updating whenever the user completes a step

A.1.3 Could have

- Landingpage
  - There are multiple version of the landingpage, for different groups of users
- The user can save his submitted data by connecting to external sites, such as Facebook or LinkedIn
- Workflow page
  - The avatar picture of the profile is linked to the user's Facebook or LinkedIn profile picture (if they connect their account)
  - If the user connects a mijn.overheid.nl, LinkedIn or Facebook account, some steps are filled in automatically with information from those sites
- There are separate workflows for users that are not a starter (for example, someone who moves from one house to another)

A.1.4 Won't have

- Features and page for the Hypofund end-product, which include (but are not limited too) the videochat, the document upload segment and the real-world mortgage advice functionality
- Steps that ask information that has been asked before, or that can be deduced from earlier steps or other context
- Unnecessary explanations that slow down the process
- Pushy UI elements that ask the user to connect to external sites
- Force the user to make an account on Hypofund, at least for the workflow steps leading to the advice

A.2 Non-functional requirements

- The project should be written in Python 2, with Django as the core web-development framework
- Every feature should adhere to the requirements

B SIG feedback

B.1 First submission

This was the feedback from Software Improvement Group (SIG), in response to our first code submission:

"De code van het systeem scoort 3 sterren op ons onderhoudbaarheidsmodel, wat betekent dat de code gemiddeld onderhoudbaar is. De hoogste score is niet behaald door lagere scores voor Duplication en Unit Size.

Voor Duplicatie wordt er gekeken naar het percentage van de code welke redundant is, oftewel de code die meerdere keren in het systeem voorkomt en in principe verwijderd zou kunnen worden. Vanuit het oogpunt van onderhoudbaarheid is het wenselijk om een laag percentage redundantie te hebben omdat aanpassingen aan deze stukken code doorgaans op meerdere plaatsen moet gebeuren.

In jullie geval zijn bijvoorbeeld er wat kleine duplicaten te vinden binnen het bestand hypofund/src/apps/users/forms.py. Het is beter om de gedeelde code naar een aparte methode te verplaatsen en die vervolgens aan te roepen. Op die manier voorkom je dat je toekomstige aanpassingen dubbel moet maken.

Voor Unit Size wordt er gekeken naar het percentage code dat bovengemiddeld lang is. Het opsplitsen van dit soort methodes in kleinere stukken zorgt ervoor dat elk onderdeel makkelijker te begrijpen, te testen en daardoor eenvoudiger te onderhouden wordt.

Bij jullie project zijn er een aantal bestanden waar configuratie in Python-code wordt gedefinieerd, zoals bijvoorbeeld src/apps/workflow/steps/step.py. In het algemeen proberen de meeste mensen een zo hard mogelijke scheiding tussen logica en data aan te brengen, vandaar dat bestandsformaten als XML, JSON, en YAML zo populair zijn. Het is beter om de pure configuratie naar zo'mn bestandsformaat te verplaatsen, om zo te voorkomen dat iemand later per ongeluk toch logica aan die bestanden toevoegd.

Het is op zich positief dat jullie ook een aantal unit tests hebben geschreven, maar de hoeveelheid tests is nog wel aan de lage kant. Hopelijk lukt het nog om in het vervolg van het project de testdekking nog wat op te krikken, zodat in ieder geval de kernfunctionaliteit door unit tests wordt afgedekt."

B.1.1 Response to feedback

The feedback said we scored 3 out of 5 score, a score indicating average maintainability. Our lower scores were for Duplication and Unit size.

To fix the duplication, we used the code inspection tools from Pycharm, especially one that found duplicate code. Using this, we fixed some of the duplication, but left small ones in place where removing the duplication would mean less readability.

We tried to address Unit Size, but in Python it is common to have some files be very large, and others very small. One of the reasons is that Python, in contrast to for example Java, can have multiple classes in one file. A lot of related classes in one file is not a definitively bad thing in Python. However, there were some instances where breaking up files indeed improved the code.

There was also a comment by SIG about our configuration being stored in code files: they were absolutely in the right there, and we fixed it by moving

that configuration data to separate JSON files, as they recommended.

B.2 Second submission

This was the feedback from SIG, in response to the second submission at the end of the project:

"In de tweede upload zien we dat de omvang van het systeem is gestegen, terwijl de score voor onderhoudbaarheid ongeveer gelijk is gebleven.

Het is goed om te zien dat de score voor zowel Duplicatie als Unit size licht is gestegen. Desondanks blijven beiden nog steeds achter vergeleken met de andere deelscores. Verder zien we dat jullie de aanbeveling van de vorige evaluatie hebben gevolgd en de configuratie code van de steps.py hebben verwijderd.

Het is ook goed om te zien dat jullie testcode hebben geschreven. De verhouding testcode vs productiecode (voor de nieuwe geschreven Python code) is 1:1.

Uit deze observaties kunnen we concluderen dat de aanbevelingen van de vorige evaluatie zijn meegenomen in het ontwikkeltraject."

Our maintainability score did not change, while the size of the system increased. The feedback also states that we implemented their suggestions for improvements after our first submission, and that their recommendations were followed in the development of new parts of our system.

C Infosheet

C.1 General info

- **Project title:** Hypofund
- **Client organisation:** Hypofund
- **Date of presentation:** Wednesday, 29th of June, 2016

C.2 Description

The project objective was to build a platform for online mortgage advice which could substitute a real-life, human adviser. The use of such a platform costs far less than hiring a human adviser, but the challenge lies in the fact that a website usually cannot give personal treatment to the user.

We researched ways to convey the feeling of receiving a personal treatment. To this end, we turned to the research fields of affective interfaces and later adaptive interfaces. This led to the development of a user interface that changed its textual content based on the user personality. To find out if this feature had the desired effect, we researched and conducted an A/B test.

To fulfill the requirements, we faced a steep learning curve of the Django framework. This slowed down development, and in the end we had to scale the test down from 200 users to 47 users. The purpose of the test shifted from expecting significant result, to a proof-of-concept test run.

Even though we would have liked significant result from the test, the A/B test served its purpose well, and it ran successfully. Running the test showed us how important it is to evaluate all properties of a feature properly, especially when its effects may be uncertain. We showed that it is possible to run an A/B test using the methodology we used, and discussed how this methodology may be improved to yield more results in a larger scale A/B test.

C.3 Project team members

- Henk Bierlee (contact: bierlee.henk@gmail.com)
  Contributions: Research, development and design
- Kilian Grashoff (contact: kiliangrashoff@gmail.com)
  Contributions: Research, development and design

C.4 Other information

- **Client:** Hypofund
- **TU Coach:** Annibale Panichella (Department of Software Technology, Software Engineering Research Group)

The final report for this project can be found at: http://repository.tudelft.nl/

D Original project description

D.1 Project description

Samenvatting: Make The World Financially Awesome

Draag bij aan een betere financiële wereld. Ontwikkel een full stack webapplicatie voor een financiële startup die consumenten ondersteunt bij het beheren van hun complete financiële huishouding: van beslisproces via productaankoop tot en met het doorlopende beheer daarvan. Ofwel de financieel adviseur, maar dan van de toekomst: digitaal en online, hoog kwalitatief, efficiënt, onafhankelijk en vooral klantvriendelijk. En dat niet 10%, maar 10X goedkoper.

D.2 Make yourself happy

Maak drie maanden lang deel uit van een startup. Pionieren, leren, realiseren. Lachen en huilen. Intensief samenwerken met experts op het gebied van (digitaal) advies, webontwikkeling, customer contact, graphic design en user experience. En voor de allerbesten: kans om langer aan Hypofund verbonden te blijven.

D.3 Opdracht onderdeel 1: Natural dialog decision support application

De opdracht bestaat uit twee delen. De kern is het ontwikkelen van een full stack webapplicatie die consumenten in staat stelt op een zo natuurlijk mogelijke wijze te interacteren met hun digitale/online adviseur. Dit betekent een vraag- en antwoordspel voor het vergaren van de juiste informatie van de klant, zijn situatie en wensen (bijvoorbeeld het kopen van een huis).

Dit betreft harde informatie zoals koopprijs en inkomen, maar ook zachte informatie zoals risicohouding en persoonlijke doelen. Deze informatie komt van de consument zelf en uit online bronnen zoals mijnoverheid.nl en social media.

De applicatie verzamelt deze info en helpt de klant de juiste financiële beslissingen te nemen.

D.4 Opdracht onderdeel 2: User interface

De gewenste applicatie is zinloos zonder een state of the art interface. De applicatie moet daarom beschikbaar zijn via mobiel, tablet en desktop. HF werkt met ervaren designers, user experience experts en customer contact specialisten die ervoor zorgen dat jullie applicatie er ook goed uitziet en functioneert.

Complex maar haalbaar

De opdracht is enerzijds zeer uitdagend. Anderzijds is het mogelijk gebruik te maken van zeer veel beschikbare tools en applicaties die reeds voor financieel advies worden gebruikt. Dit laatste zorgt voor veel betere realiseerbaarheid.

De uitdaging zit in het met elkaar verbinden van de klantsituatie en de (wettelijk) noodzakelijke informatie voor deze adviestools om te komen tot de beste oplossing voor de klant.

D.5 Aandachtsgebieden

Pattern recognition, natural speech, neural networks, robo-advice, rapid prototyping, fintech, natural dialog generation, text recognition, scalability, behavioural finance, machine learning, data mining, security, privacy, gamification, ...

Ofwel: alles wat nodig is om een eigentijds systeem te ontwikkelen om consumenten snel en goed te begrijpen en van de juiste feedback te voorzien.

D.6 Wat wij bieden

Intensieve kennismaking met een financiële start up. Begeleiding door ervaren software- en systeemontwikkelaars. Samenwerking met user experience experts en graphic design-veteranen. Een integraal beeld hoe technologie, financieel advies, business development, commercie en klantgerichtheid samenkomen. Persoonlijke feedback.

Desgewenst een inspirerende werkomgeving op ons kantoor in het centrum van Utrecht.

Twijfel of meer weten? Neem contact op met een van de initiatiefnemers.