Dublin Business School

School of Business

Master of Science

Information Systems with Computing

Design, Implementation and Evaluation of an Innovative Hybrid Support System

Mario Cuba

August 2015

Acknowledgements

This piece of work could not have been possible without the knowledge, guidance and support from my supervisor, Michael Gleeson. Since DBS's open day until the very moment I wrote these words, there has been nothing but words of encouragement and the inspiration to do the best possible job that I can. This whole year has been exceptionally difficult but it was made easier by him. For that, and much more, I'll be forever thankful.

I would also take the time to not only thank, but to dedicate this effort to my family. Being away from home to pursue my career goals has been very difficult for all of us, but they have once more put my interests and wellbeing first. They have made this whole experience much easier than I anticipated, which only gives me more reasons to strive to repay their love and support someday, somehow. I hope this makes them proud.

I also want to thank my lecturer Alan Graham for his teachings. I have been in countless classrooms with an innumerable amount of teachers, and I had never encountered someone with his contagious passion. Writing this document was so much easier thanks to him. "Write a paper that makes your grader say "Jesus, I can't deal with this today". You want to be that paper." Sorry, Michael.

Finally, to everyone that participated in the test and data collection stage of this project: you have made this worthwhile, and I thank you for that. In such a stressful period with tight deadlines, it's good to take a step back and remember that you always have to find joy in what you do, no matter what.

Declaration

I certify that this research, submitted to the Postgraduate Business Programme Coordinator of the Dublin Business School in Dublin, Ireland, is entirely my own creation unless otherwise stated and it has not been taken from the work of any other author.

This project has been prepared and developed according to the regulations and specifications defined by the Dublin Business School module descriptor for the Award stage, and it has been submitted for evaluation with no other purpose than the fulfilment of said academic requirements.

Signed,

_____

Mario Cuba

Date: August 21st, 2015

Word Count: 9338

Abstract

Support teams are usually the first personal encounter a customer will have when dealing with a company, therefore, this can make or break the relationship they may have with each other. Nuances make the difference: Was the agent knowledgeable? Did he treat the customer with care? Did the system put too many hurdles to get support? Is there enough information available? All of these questions will go through a customer's mind when gauging the quality of the support they have received and the levels of satisfaction they feel after their experience.

Nonetheless, one of the most overlooked factors that drive customer satisfaction is how do customers get in touch with the company, and what methods are made available. Should live chat be used? Or maybe just an email? Would a phone call be the right answer? Without noticing, using the wrong support channel may lead to a poor experience, even if the aforementioned nuances have been properly addressed.

This is what this research document is all about. By using modern web technologies such as Ruby on Rails and Node.js, a proof-of-concept application will be developed and put to the test by real-world users of these systems. By evaluating the results of a test drive of the prototype application, it will be possible to dig deeper into customer's minds and answer the question: will switching back and forth between methods of support—in this case, synchronous (real-time) or asynchronous— using an integrated solution, provide a better experience overall for the customer, depending on its situation?

Table of Contents

Chapter 1

Introduction

Customer satisfaction is paramount in today's exceptionally competitive online world, where businesses struggle and race against each other on a day-to-day basis to provide the best customer service they possibly can. The advent of technology in this field has made a significant impact in the perception that customers and clients have, where no solution is far-fetched or timely enough.

1.1 Background and Brand Loyalty

The constant evolution in the world of Information Technology, or IT for short, allows the smallest of start-ups to the biggest corporations to provide services to their customers in new and creative ways, allowing them to gain an essential edge in the market: brand loyalty.

Brand loyalty has several beneficial effects for the company in customers. A study by Gaur et. al (2014) found that, amongst others, "longer tenures" and "lower sensitivity to prices" were in the top behaviours found when brand loyalty is high. Taking this into consideration, fine-tuning the customer experience since the beginning —where a customer first inquiries about services or has an initial problem with a platform— until the very end —when the customer is finished with the service or the product, and therefore, the experience— should be seen as a highly profitable investment in the long run.

1.2 How Do Companies Provide Support?

Regardless of the product, service or field, customers expect an impeccable support experience and this is delivered in an assortment of ways. Some tech-centric companies, such as Envato Ltd. (the Australian company behind ThemeForest), focuses exclusively in providing help to their customers, including copyright infringements, payments and account issues, using ticket-based online systems. Others, such as Stripe Inc. (the Irish payment provider start-up), uses emails and IRC to provide technical support, sales enquiries, and other tasks. The variety does not stop here — other companies like Bank of America rely primarily in phone calls to provide different services to their clients, but they also provide several online options.

Clearly, a pattern emerges. Companies do use varied methods of communications to support their users. Some rely on one, and some in several standalone tools. This begs the question: would using an application that combines these paradigms be well-received by the customers, increasing customer satisfaction?

1.3 Project Aims and Objectives

This study seeks to determine the benefits and drawbacks of using a hybrid ticket-based customer support system by providing a proof-of-concept prototype that will allow both synchronous and asynchronous communication between agents and clients using a responsive web application.

Prior to achieving this, a series of goals must be reached. These are:

- Compare different methods of communication used by various ticket-based customer service information systems.

- Establish the technologies to be used in a proof-of-concept prototype that will enable real-time and asynchronous communication seamlessly.
- Identify major areas of concern and potential improvement in common scenarios where ticket-based customer service systems are in use, both in the prototype and in general.

Chapter 2

Literature Review

2.1 Synchronous Communication

Real-Time or "synchronous" communication is a paradigm where a group of people as small as two can address each other instantaneously and "is aware of the presence of the other" (Schwartzman, 2013), regardless of the medium, which in the context of this research, is text-based.

There are companies that favour this sort of communication, as it is argued that it is a determining factor in areas like sales. A study made by Forrester in 2010 indicates that 44% of customers consider that using live chat whilst shopping is "one of the most important features a site can offer". It is worth noticing, however, that this has a *sales* impact and not precisely a *customer satisfaction* impact, since a feature like this can make or break a sale, but it may not be considered as fundamental in other contexts. It can be inferred that it is a conversion tool more than a support tool used in this way.

Other companies do focus on the customer support side of things by using live chat, with good results. It has been found that customer satisfaction ratings can rise up to 92% when implementing proactive live chat for the first time in a company — the highest of all the channels available, including voice, web forms, emails and social media (The Zendesk Benchmark Q1/2015, 2015).

Combining these two results and several others that reaffirm that success can be achieved implementing live chat in a proactive and

thoughtful way. If this is the case, then why not all companies implement this sort of channel for their customers?

2.2 The Drawbacks of Real-Time Communication

The Zendesk Benchmark Q1/2015 also indicates a key aspect of not only live chat systems, but any other synchronous services: an inverse correlation between the number of live chat opened in a given month, and the customer satisfaction ratings. The higher the number of chats, the lower the satisfaction rating.

Taking this into consideration, context becomes more and more important. This is why sales in an e-commerce site (which are characterised by shorter, to-the-point conversations) are more positively received and better handled than, for example, IT or legal departments. The larger the and more complex workload, the more customer satisfaction suffers — which leads to more "traditional" support systems.

2.3 Asynchronous Communication

Bertram et al define Issue Tracking Systems as tools that manage a variety of activities focused on archiving information which is used in tandem by "numerous stakeholders throughout the lifecycle of the software" or project in a broader sense. For the purpose of this research, Bertram et all definition has been adapted by the author's own experience in the area of customer support, adding that they are structured alternatives to real-time communication in the realm of customer support. These provide tools that ease and speed up the process of problem-tracking and resolution in an orderly fashion, which is often transparent to the customer. They function in an asynchronous way — that is, the customer or end user does not need to be in presence of the agent to describe their problem and ask for

help, and at the same time, the agent can receive the request and adequately handle it a reasonable amount of time later.

2.4 Differences Between Both Approaches

There are significant benefits of using this mechanism to engage customers and to handle their issues and concerns. The most important one is that the software, not the agent, will remember everything for them, keeping a detailed history of the actions taken by both the customer and the agent, which will allow anyone that will need to be aware of the situation to get up to speed efficiently and effectively.

This does not only apply to customer service only — Joel Spolsky, creator of Trello and Stack Overflow, asserts that tracking software is "one of the hallmarks of a good software team" and that "one of the biggest incorrect facts that programmers consistently seem to believe is that they can remember all their bugs or keep them on post-it notes" (Spolsky, 2000). This is a key feature that live chat software often lacks, and since both services usually work in parallel to each other, it is likely that more problems than solutions will arise. Customers notice this, and satisfaction is affected in return.

Another differentiating factor between both methods of contact are waiting times. The threshold between what is a reasonable amount of time for a response and not is much less in real-time communication than in an asynchronous environment, particularly when dealing with larger, more complex issues. To give a simple example of this, it would be reasonable to wait two or three minutes for a response to a live chat, and the same goes for 12 or 24 hours for a ticket response. Waiting in line for 12 hours to chat with a customer service representative, however, would be ludicrous. Companies know this, and take

advantage of this phenomenon to balance their workload, make decisions, deploy personnel and deliver solutions to customers.

2.5 Usage in this Project

With this knowledge, it is possible to devise a solution that will organically switch back and forth between modes of communication that would suit both the customer and the agent, according to the customer's situation and the agent's availability. This process should be mostly transparent to the user, and should eventually increase customer satisfaction, because the experience should be suited to the level of urgency, the type of issue, and the availability of both the customer and who is helping them.

Chapter 3

Research Methodology and Methods

3.1 The Object of Study

This research project has been conceived primarily by the author's own observations whilst labouring in the world of customer service for the past 8 years, by attending to different types of population under different circumstances.

The phenomenon, which is the constant feeling of frustration amongst customers when they are both waiting long periods of time for answer to simple queries and how several systems in place (live chats, issue tracking, social media, etc.) are not in sync with one another, leads to the design of a prototype that will switch back and forth between two modes of communication —synchronous and asynchronous— so the customer has clearer expectations from the onset of the situation and does not have to choose between a method that is probably both inappropriate for neither them nor the company they are trying to get in touch with.

3.2 The Research Philosophy

This is a practical solution to a practical problem in a real-world scenario. As clearly demonstrated in the literature review section of this document, the tools do exist but they are not used in conjunction with one another.

Instead, they are frequently used independently to solve different problems, but they are seldom used in tandem to provide a

consistent, organised, and equally productive solution to customers in need of expertise and assistance. This research will try to either prove or refute the hypothesis that using the correct mode of communication given the circumstance in an integrated system can increase customer satisfaction.

3.3 Gathering Data

Due to what has been mentioned before, it is mandatory to get true insight from potential real-world users of support systems, which are found virtually in any area of day to day life, leading to a primarily qualitative research.

For example, one study found that 77% of global users shopped for electronics online (Statista, 2014), and another one indicates that 78% of the population of the United States alone purchased something online in the first 3 months of 2014 (Smith, 2015). It is possible to infer from this data that all of these customers have or have had access to a support system of some sort: to ask questions to an agent, to look for answers themselves, to be guided through the purchase flow, to ask for recommendations or solve a last-minute issue.

Moreover, another study estimates that in the United States only, around 201 million people are digital consumers and have shopped online or will do so during the course of 2015 (Statista, 2015), which means that almost two thirds of the whole population of the country (United States Census Bureau, 2015) are online purchasers currently and growing rapidly, around 15% on 2015 (“Ecommerce Growth Benchmark”, 2015). Therefore, it is possible to deduct that they too have access to a form of support system, making them eligible for this study.

To add to the sample of people that could provide valid feedback about the feasibility of the tool used to conduct this research, it is necessary to include subjects that are not necessarily online shoppers, but do use some sort of online service that provides online support.

Examples of these services include music streaming (Spotify, Rdio, Apple Music), online banking (AIB, Bank of America, Barclays), single and multiplayer video games (World of Warcraft, League of Legends, Steam), government or other public services (Garda National immigration Bureau, Internal Revenue Service, Her Majesty's Revenue and Customs), booking services (AirBnB, British Airways, JustEat), technical support (UPC, Dell), regular mail (DHL, UPS, FedEx), mobile service providers (Vodafone, Telstra, Movistar), even e-commerce shops that answer questions (Amazon, eBay, etc.) and many others — all of these provide some sort of synchronous or asynchronous customer support, even if it is as basic as email-based, yet it is not necessary that the customer per se had to go through an online purchase or acquired a product that needs to be purchased through an ecommerce site to have access to these facilities.

By taking this into consideration, it can be concluded that virtually any person that has an online presence of any degree is more than likely to have had contact with even the most basic form of a support system, which makes them eligible as a sample for this research.

3.4 Sampling and Distribution

Knowing the broad spectrum of individuals that can be eligible for the study, a proper channel needs to be used to both disseminate

the experiment and that allows simple random sampling without replacement.

In this context, this means that individuals can both use the application and provide feedback several times at their will depending on the type of experience they had and the level of detail and depth they have studied the artefact.

The internet, and the sub communities that live within it, are the perfect spot to take a random sample of subjects that are willing to participate in the research. This study uses high-traffic sites such as Reddit, Facebook, or Twitter to get individuals to participate — which allows the experiment to naturally disseminate.

3.5 Previous Researches Using Similar Methods

This is certainly not the first time that research has been conducted in open internet communities such as the ones previously mentioned. A prime example of this is located at the University of Illinois at Chicago, which has already undertaken a health-related research project targeting tobacco-related attitudes, beliefs and behaviours by harnessing the power of social media sites and forums.

In their article, “Shout-out: Are You Using Reddit for Public Health Research? What Works?”, they mention that they have worked with data taken straight from YouTube, Facebook and Twitter, and they aim to use Reddit as well due to “the user composition, content, format and usage norms” which sets them apart from the rest (Buenger, 2014).

There are other studies in different fields of science as well, leveraging the information that is contained within those sites and by using their users are research subjects, such as a 2014 study by Ph.D.

student Hannah Bowers about emotion processing and IBS in the Royal Holloway University of London, where she encourages other Ph.D. candidates to use Reddit as a primary source of data "so long as you have a thick skin" ("Using Reddit for Research", 2014).

3.6 Primary Data

The primary method of data collection will be a mix between an experiment and a case study, followed by a questionnaire to gauge the satisfaction of the user after using the proposed system.

The experiment side of the research is an actual run through the system, where they will be able to submit a ticket, receive a notification, then wait for instructions on how to continue. After this, they will be able to log into the system, open new tickets, look at their history, and join live chats when an agent, which is the author of this study in this case, replies to their inquiry. After the issue has been closed, the user will have sufficient knowledge to provide their feedback about the system through a 5-question questionnaire focusing on the ease of use and feature content of the system. This information will be integrated into the prototype at a later stage.

The data gathering side of this research has been carefully devised in a way that it is not overwhelming to any of the participants. Since it is mandatory to use the system at least in a superficial manner to provide valid and significant feedback, the questionnaire was purposely made short and concise, including only the relevant pieces of data needed to further the progress of the development of the application and to get closer to an answer to the research's hypothesis.

Chapter 4

Artefact: Design and Implementation

4.1 Selected Development Methodology

In every software development process, regardless of the size of the team or the project itself, it is of utmost importance to have a clear understanding on how to tackle the problem ahead and a proper plan to do so. This research undertaking is no exception.

Software development methodologies help to achieve success when designing and writing applications of any type. These propose strategies and an overall framework to organise teams, individuals and tasks in a certain fashion, depending on the selected methodology. Said methodology should be meticulously studied so it can fit the team's culture, skill, the project's scope, timeline, and reach.

After careful consideration, it was decided that using the Agile Software Development methodology would be ideal for this project, using a modified Kanban approach that would fit a single developer.

The Kanban approach, made famous by the Toyota Motor Company in 1940, focuses on flexibility, visual process management, and "work that's actively in progress", although if required, it is possible to re-prioritise work to better suit the current requirements (Radigan, 2015). This ensures that, by the time that the developer finalises a story, they deliver the maximum value to the stakeholders, the end user, or the company.

This framework is composed by a board that displays all the user stories available to work with, the current stories being worked on, and the relevant information to process them. Within that board, there is a backlog of user stories which serve as tasks that need to be completed in order to further advance the project, and other metadata such as labels or in-depth descriptions that will help to quickly identify the issues at hand and the necessary information to process them.

Since the conception of the application, a backlog of tasks that needed to be accomplished was gradually being filled with user stories with the appropriate template of "As a <role>, I want < desire> so that <benefit>", common in Agile environments. This would allow the author to have a birds-eye look at the tasks that needed to be done, with the appropriate context, eliminating guesswork later in the development process.

Some elements common in Agile environments, such as Test and Behaviour Driven Development, Pair Programming, Time-boxing or Retrospectives were left out of this project due to the size of the team and the scope of the project, as they were deemed not mandatory and would not provide as much value as desired, at least in the early stages of the application lifespan.

By using this methodology, the author has ensured that all the necessary tasks were completed before the required deadline, along with the accompanying document and relevant research mechanisms and data.

4.2 Technologies Used in the Asynchronous Side

Given that the application in this research project is divided in two main components —a server-side application that is used as the asynchronous part and complementary client-side application that is

used as the live chat— it was necessary to use different technologies that were suited for the task.

On one hand, the programming language Ruby was selected to develop the server side, asynchronous application. Ruby is a dynamic, loosely-typed, object-oriented and general-purpose scripting language that is frequently used to develop high-performing web applications, without sacrificing developer productivity or enjoyment, which are the language's primary purpose according to its creator. (Matsumoto, 2008).

Ruby is also one of the most used and requested languages in the world, according to the Stack Overflow Developer Survey 2015. It occupies the 11th spot next to other extremely popular languages such as C, Python and PHP ("Stack Overflow Developer Survey 2015", 2015).

Aside from these facts and benefits, one of the author's main objectives with this research is to hone his skills in this programming language by writing the application from scratch, which made it the perfect scripting language for the task.

For the purpose of this research and the development of the application, the latest stable version of the language, 2.2.2, will be used.

This is accompanied by Ruby on Rails, the most popular web framework in use within Ruby programmers ("Web App Frameworks", 2015). It provides a plethora of tools to speed up the development of both simple and complex web applications, without compromising the security, robustness, or technical capabilities the software will be able to provide. These benefits and other features will be explained more in

depth under “Architecture”, later in this chapter. As for the version that was employed, Rails 4.2 was chosen, as it is also the latest available version.

Other tools that were used were Twitter Bootstrap, which is a frontend framework to speed up the development of user interfaces by providing a comprehensive set of ready-to-use elements using CSS and JavaScript. MySQL 5.5, a relational database management system served as the primary data storage. Nginx, a web server software, was deployed to be used as a reverse proxy to handle high-latency inbound connection attempts, to serve assets via HTTP and to enable the usage of node.js, which will be explained later in this chapter. Finally, another server, Unicorn, was used to serve low latency and high-bandwidth users as the main response handler.

Gems, which are the equivalent of libraries or packages in the Ruby world, were also used in this project to an extent. It is worth noting that Ruby on Rails itself is a gem, and it is dependent of several other.

Some non-standard gems that were used in this project were Faker, which enabled to populate the database tables with real-looking, but ultimately fake data programmatically each time that the schema was set up via rake —Ruby’s “make” program— while still allowing real data to be entered.

Figaro was also used, which enables the usage of an individual file encapsulating all environmental variables that are needed by the application in a single place, increasing privacy, security, and the process of committing to projects. This is especially important due to the fact that the application is available publicly in GitHub.

4.4 Technologies Used in the Synchronous Side

On the other hand, the real-time facet of the application was developed using tools and technologies that are significantly different to the ones previously discussed. Given the synchronous nature of the application, it was decided that the node.js runtime environment would be ideal for the development of the application, due to the fact that the event-driven architecture and I/O capabilities of the environment fit perfectly within the scope of this project.

Node applications are mostly written in a mixture of JavaScript, HTML and CSS. This part of the application was not built from the ground up, but adapted from a freely available live chat script published by TutorialZine under a license that allows the modification and use of the code in a context such as this. The integration with the Rails, however, was coded independently.

Node has a similar packaging structure than Ruby. It uses packages that are installed, configured, and distributed by npm, which stands for “Node Package Manager”. To be fully able to perform the tasks that are needed to achieve the success of this research, packages such as Socket.io and Express.io were used extensively.

Socket.io is an event-driven JavaScript library, just as node.js, used in real-time web applications. It is an essential part of this project, as it provides all the synchronous capabilities, enabling techniques such as bidirectional communication using WebSockets (an HTML5 technology), both in the client and in the server side.

Express.io is a compliment to Socket.io. It is a web framework, much like Rails in many ways, that facilitates the development of web

applications using Socket.io. It handles tasks such as serving assets, rendering templates, routing, and more.

4.5 Architecture

Given that Rails heavily relies on the "Convention Over Configuration" philosophy, many of the design decisions that were taken in this project are an extension of Rails' way of development, to speed up development time, reduce the number of decisions that need to be made and increase simplicity.

The application uses a Model-View-Controller architectural pattern, inherent in Rails. This patterns allows developers to achieve a higher degree of separation on concerns, detaching the user interface (the View) from the application logic (the Controller) and the business logic and data domain (the Model).

All requests go through a reverse proxy configured to receive and buffer all data that's sent to the server by using nginx. The main purpose of this is to lower the latency in each request that is made and to serve static files, leaving Unicorn, the app server, to deal with the more processor-intensive tasks, all which are being made by Rails. Unicorn is set up as a service at start up time so the operating system always tries to invoke the server in case of a crash or a reboot, keeping uptime to a maximum.

The requests come from the views and they are sent to the controllers. The controllers will then decide how to best handle the request and call the models if this is needed. The application is designed to be RESTful (another Rails convention) so it natively supports different types of responses. Currently, it supports receiving and transmitting responses in HTML and JSON, the latter being

particularly important when taking with the synchronous side of the application.

The real-time chat also receives all the requests via nginx. Node.js is configured to run on port 8080 using PM2 (a node process manager) so it is properly handled by the operating system in case of crash or reboot, by daemonising the running application, just like Unicorn.

All of this stack is on top of an Ubuntu 14.04 LTS installation, provided by a Digital Ocean Droplet –which is essentially a VPS- with dynamically assigned RAM that suits the current load the system is handling.

For development and deployment, Git is used. The way this works is that the main files of both sides of the application are pulled from the main public repository, hosted by GitHub, then a new production branch is created. After this, all changes committed to the master branch are then merged into the production branch to ease the resolution of merge conflicts then the Unicorn worker is restarted to apply the changes. In later stages of the application though, a more complex and robust method of deployment using Samson and Capistrano will be implemented, automating this whole process.

4.6 Class Diagram

Proper, careful and considerate planning is mandatory when designing a system that needs to be written from the ground up, especially when dealing with software teams.

When speaking about design, Class diagrams are the building blocks of any piece of software, since they paint a clear picture of the entities that need to be developed, and how will they be

interconnected, making writing the software a much easier, robust, and faster task.

The following class diagram deals exclusively with the classes that were written specifically for this project, since the framework itself includes an abundance of classes —both for the proper functioning of the framework and for the developer to use— that are not necessary to show.

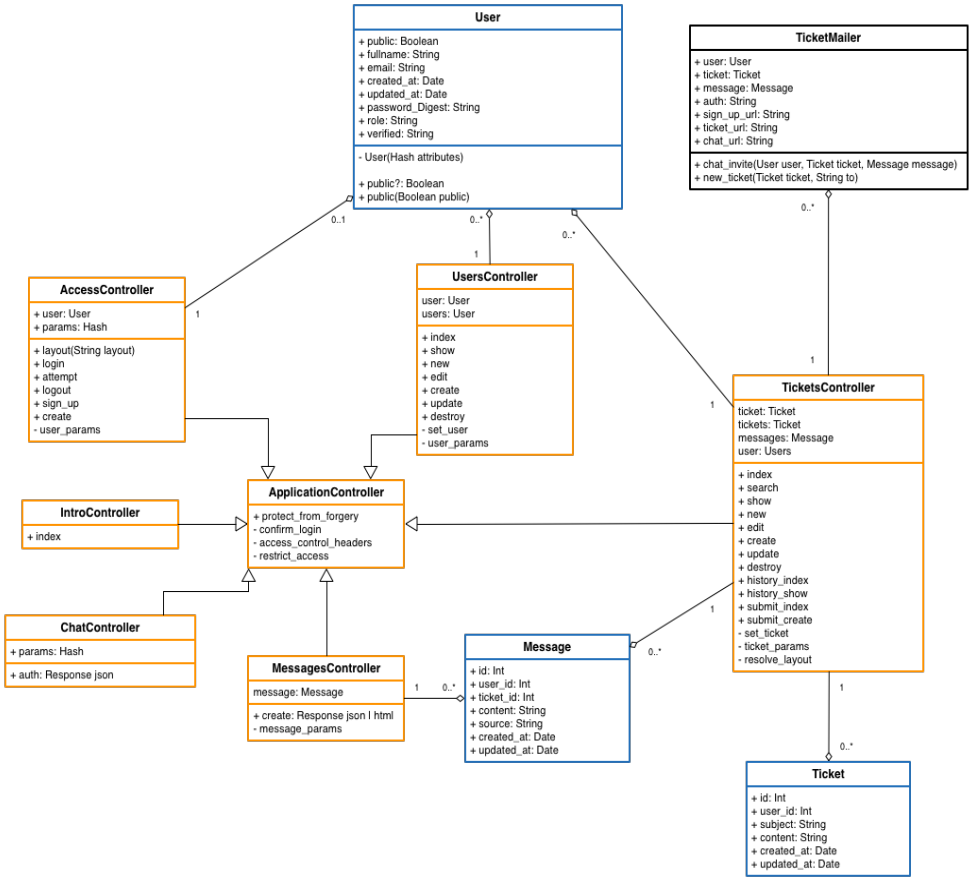

Fig. 1: The Application's Class Diagram

4.7 Entity-Relationship Diagram

Rails uses ActiveRecord as their Object Relational Mapper, which is in turn implemented in all the models in the application. This means that all the database interactions are strictly performed by the

models and nothing else. This is tied to a notorious philosophy in the Rails world: “intelligence belongs in your models, not in the database” (RailsGuides, 2015), meaning that features like database constraints, validations, and referential integrity is exclusively handled by the models, not the database.

The following entity-relationship diagram will show the application’s database schema that’s automatically generated by the Models according to the previous Class diagram.

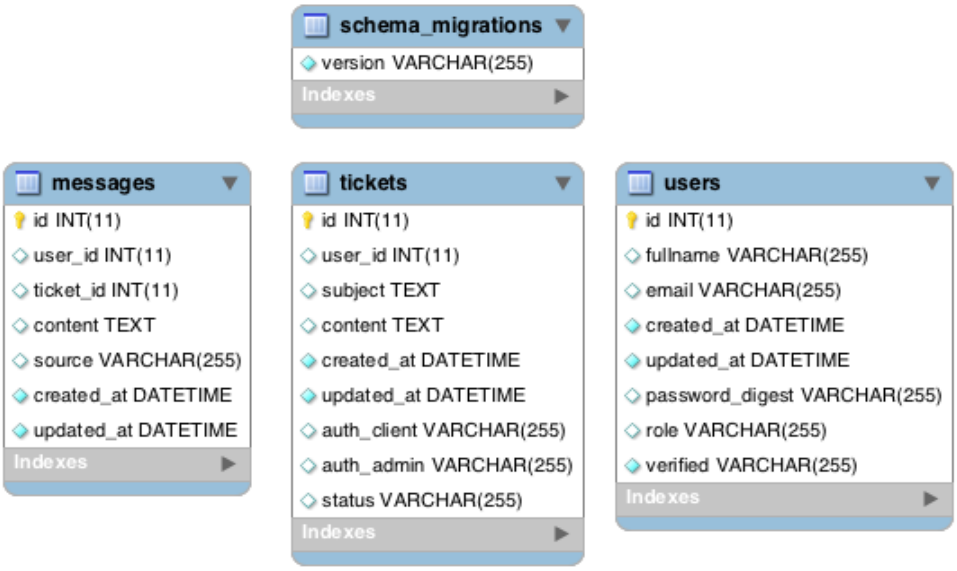

Fig. 2: The Entity-Relationship Diagram

Chapter 5

Research Findings

5.1 Data gathered

As mentioned in previous chapters, the research was conducted by setting up a live test environment with a running proof-of-concept available for any user. The application, which resides on relayapp.net, was distributed in an assortment of ways to be able to achieve the necessary amount of exposure, and therefore, the appropriate randomised sample size. These methods include social media outlets such as Facebook and Twitter, online communities like Reddit, and other groups of people, such as a local language school.

Of over 50 people that came across the experiment and interacted with it in one way or another, a total of 26 respondents took active part in it, surpassing the previously set goal of 10. The average time to complete was 10 minutes, which was also exactly the amount of time the author of this research was aiming for, due to the additional time the subjects needed to interact with the application. This also allowed a greater amount of detail from the participants without being too overwhelming.

The feedback about the application was also delivered in various ways. Some of the participants were keen on providing it through Typeform –the tool the author provided to the subjects so they could provide their feedback via a questionnaire– and some others felt more appropriate to have the conversation directly through the application. Both types of participants provided relevant information

that would prove to be useful for the further development of the application. In the aforementioned questionnaire, only 5 questions were asked with 2 being quantitative in nature. The questions are:

- Is there anything you liked about the application?
- Is there anything you did not like about the application?
- As a customer, what features would you like the application to have?
- Would you prefer to have this approach in your next customer service experience?
- Was the application easy to use or hard to use?

To gather real, usable and accurate data, the author proposed to participants to send any sort of query through the system, including questions about the author, the research, about their tasks and obligations, hypothetical scenarios, or casual conversation. This would allow users to have a “proper” customer service experience and would permit them to use the system to the degree they needed or wanted to. Only basic pointers were given and then the users would figure out how to use the application to gauge the ease of use.

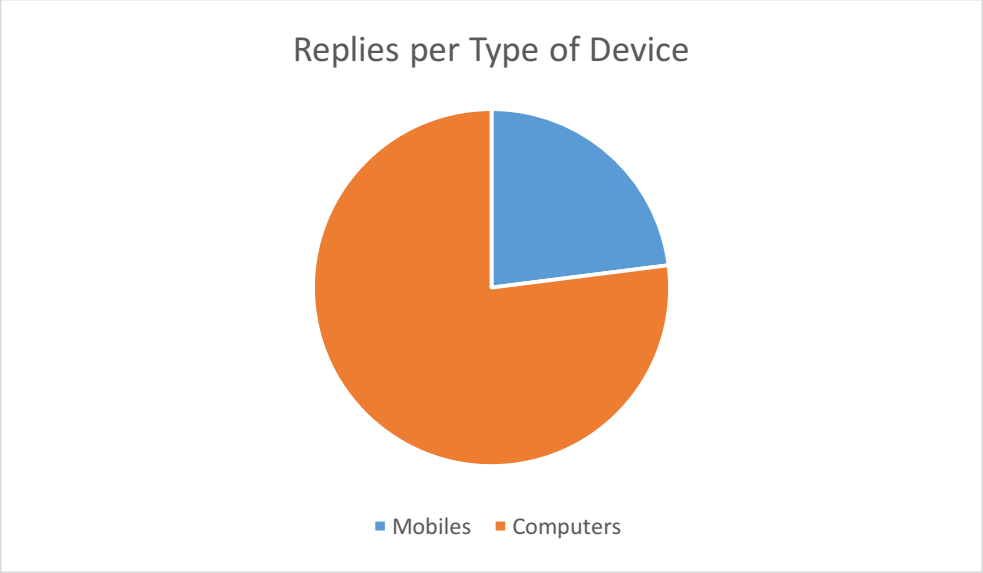

Fig. 3: Participant's usage of devices when replying to the form. Mobiles include tablets and smartphones. Computers include laptops and desktops.

As for the stats of the results, they exist only for the respondents that used Typeform: 23% replied through a mobile device, whereas the rest of the participants used a desktop or laptop PC to respond. Given that the application is on its early stages, usage analytics are not available, although this does not affect the study in any way.

As for the feedback provided in the application itself, it came from both channels: the synchronous and the asynchronous ones, which will also be taken into consideration for the next section of this chapter. Furthermore, for the purposes of this research, feedback that was provided outside of the application and the questionnaire, but matches the questions asked in the data gathering method, will be taken into consideration. The author theorises that feedback was also provided this way due to the technical knowledge of certain participants.

5.2 Feedback and Improvements

This section will discuss individual pieces of feedback provided by the users of the system, alongside some discussion and insight about it, plus implementation details.

5.2.1 Overall design

The first piece of feedback that was provided about the application was design-wise. Although the aesthetical aspect of it was praised at first since the application uses a material design-inspired theme provided by Bootswatch, a user made the observation that the application uses a significant amount of whitespace, which results in eye sore after prolonged exposure, especially in high-contrast or high-brightness monitors.

Upon further research, some high-profile sites such as Facebook or Tumblr try to reduce whitespace significantly by using darker colours in their backgrounds, then highlighting information with white backgrounds. Furthermore, the official Material Design spec guidelines provided by Google state that the colour of the background in applications and sites should be a light shade of grade to significantly reduce this problem. White is only used in dialogs, cards, and un-accented menus, such as sidebars.

To counteract this, the application was updated to have a darker background by adding a light shade of grey except in dialogs and menus, as per Google's specification.

5.2.2 Status buttons

In the original design, the "Close" status button, which sets the status of the ticket to "Closed", which in turn, removes it from the "All tickets" view, is coloured blue –the same blue used in the navigation–

and this is the cause of confusion. Although this was originally intended to be a call to action so users would know that they could press that to solve their issue, it was confused with the actual status of the ticket, regardless of the fact that the actual status is being shown a few pixels before.

A few solutions could be implemented. One of the suggestions was to the use icons to symbolise what the button does instead of signalling that it is a way to check the status of the ticket. Another one is to remove the colour altogether, so they all look like a button and does not transmit any other kind of information. Last, but not least, would be to use a Single Button Dropdown or a Split Button Dropdown as provided by Bootstrap, so the main button conveys what the button does (change the status) then the dropdown would show which buttons to use.

The application was updated to use the latter option, as it seems to be the most appropriate for the situation.

5.2.3 Email Validation

One of the users found out that the client side validation for emails was not working in the submit form within the application. The application was designed to prevent this, but not all forms had this feature implemented. The feature, which are HTML5 form fields, were missing and eventually added to the fields that needed it.

After the fix, the issue continued to happen in Safari –one of the modern browsers that are supported– which led to unexpected data to be entered into the system. This also brought to the surface a few more bugs. The reason this was happening is because Safari does not enforce

client-side validation for “email” type input fields, unlike Chrome, Firefox, or Opera.

To prevent issues with the application, the model validation was updated to match the other browser’s validation rules, which are:

- Email must contain an “@” sign
- Email must have an extension

5.2.4 Support for Gmail aliases

Users have different ways of handling their email address and Gmail provides a few interesting methods to do so. One of them the usage of aliases, which allows a great deal of flexibility for inbox management.

The way that this works is by appending a plus sign (“+”) and a name for the alias to the end of the email address, before the at sign. All emails sent to that address will end up in the original inbox, and it can be filtered as an independent address. For example, if the original address is mario@gmail.com and this address needs to be used for a game’s announcements, mario+game@gmail.com could be used and all email will be received without a hitch.

A user tried using their email alias with the application and it worked successfully, but had issues initially when trying to log into the live chat. This is because the link that is provided by the application, which contains both the email and the authentication code, does not support plus signs, contrary to what the RFC 2822 defines.

A solution to this issue is to allow plus signs by modifying the regular expression that handles taking the URL of the application and

setting the proper variables. This will allow users with Gmail aliases to have a smoother experience when connecting to the live chat.

5.2.5 Notification messages

The application sends a confirmation message to all the users that match the following criteria:

- User creates a new ticket
- User receives a reply from an admin

This confirmation message contains the following information which is necessary for the user to know where to go next in case they need further assistance:

- The ticket ID
- The content of the original message or the response
- A direct link to the live chat application
- A direct link to the ticket inside the system

One of the pieces of feedback provided about this message was that, if it was a multi-line message, which is supported by the application, line beaks would not appear in the notification message. This means that if the message is anything but short, it would be very difficult to read. This was fixed in later revisions of the application for the comfort of the users.

Another piece of feedback received about the message was that, in the HTML version, the authentication code was not being shown in an intuitive place. The link was located in a list and the authentication code was located outside of that list, which makes it feel like that information belongs to another section. This was also fixed in a later revision.

5.2.6 Live chat

There were several bits of feedback around the live chat, which was overall very well received. Most of it was centred about its design and some tweaks that could be beneficial for the users.

The first piece of feedback provided was around the size of the messages sent and received. The fundamental issue with how messages are displayed on-screen is that it shows an avatar of the sender and the receiver per individual message.

The alternative to this would be to group messages per sender. For example, if person A sends 5 messages, only one avatar will be shown. Person B replies with 3 messages, and only one avatar will be shown as well. Person A replies once more with one more message and another avatar will be shown since the previous group is Person B. Widely known and used chat applications such as Facebook Messenger and Google Hangouts use this approach.

Another important bit of information provided by the users is somewhat related to the previous one. Established chat applications industry wide, like iMessage, WhatsApp Messenger, Telegram, and others, position outgoing messages to the right of the screen, while incoming messages are positioned to the left of the screen. The application took the opposite approach, which is confusing to users.

Finally, a cause of concern for users was the lack of responsiveness of the live chat, even though the asynchronous side was developed using a mobile-friendly framework and it does support several viewports. In mobile devices, a simple desktop-only version was served which made difficult for mobile users to use the chat. It is still totally functional, but very difficult to use, especially in low-end

devices. This is a serious problem that was only found in the later stages of development, since making the live chat responsive was one of the early goals of the proof-of-concept.

Due to strict time constraints surrounding the project, it is not possible to make this modification to the application, although it will be implemented in a future patch.

5.3 Features to be Implemented

In this section, suggested features by both the author and the customers that used this application will be discussed in-depth.

5.3.1 Markdown Support

This is functionality that as soon as the proof-of-concept went live, both the author and several users noted that was missing. The application is by nature communication-centred and additional tools are needed to properly convey messages in a clear and easy way. Markdown is the answer to this.

In the first stage, Markdown would be implemented in the asynchronous side of the application, in every free-text field (i.e. ticket creation and responses). This would allow elements like links, lists, quotes, and other text formatting to be used within the application and outside of it with minimal setup and effort from the user, given its easy to use syntax.

5.3.2 Screenshots and Other Attachments

Following up on how the application is specifically designed to be a communicative medium, sometimes it will be necessary to use more than text.

One of the suggestions was around the submission of screenshots to illustrate a potential problem, which is crucial in the troubleshooting stages of an issue, but it could be more than this, as service may not be limited to images. For example, it may be required by a user of the application to send a document for the Company that employs the application, such a DMCA take down notice.

The particular case of the screenshot is a bit more complex because it may require additional and external tools, like Droplr or some bundled applications like Window's Snipping Tool. Assuming that the file transfer is what is important, a drag and drop interface could be implemented that would allow common file types such as Zip or PDF to be uploaded. It can also be possible to use 3rd party services like Google Drive or Dropbox to facilitate file sharing.

Another solution which could be implemented faster, akin to an MVP (minimum viable product), would be to implement the API of a service like Imgur to simplify image hosting and sharing, in a way that files are stored securely without taking up hosting space, located inside a user's profile, and easily editable within the service's website if needed.

5.3.3 Agent Profiles

Another suggestion from a user is around agent profiles or bios. The logic behind implementing these is that, depending on the problem domain of the company who is using the application, a certain degree of information about the person who is taking care of the situation should be available.

This is especially important when handling complex issues like legal proceedings, where a knowledgeable expert is an asset, or when

providing support in an environment where several areas of knowledge are handled, like a research-focused university, where experts of an assortment of branches are available and users are looking for one in particular.

Considering that user profiles already exist within the system, the functionality could be extended to include additional information, and it could be made available to all users who log into the system or that are having a live chat session, which also requires a degree of authentication.

5.3.4 Custom Fields

Every organisation has different needs, so applications strive to fulfil as many they possibly can. This can lead to two possible results: users are happy because of the flexibility the application offers, or users feel overwhelmed because of the abundance of choices and could potentially deem the application as bloated or overly complex.

It is worth noting that this particular piece of feedback was an idea, more than a request. This user, and many others, did mention that one of the strongest points of the application was its minimalism, which will be discussed later in this chapter.

With all this in mind, it is a feature that could be implemented in the future, but the application will remain providing streamlined functionality in the meantime.

5.3.5 Authentication Methods

Users also appreciated how straightforward the process was to create a ticket in the application. This was an effort to eliminate any hurdles that would prevent users from getting in touch with agents and

reduce the complexity of the application in general. A key aspect of this is the fact that it is not necessary to create a user to neither create or retrieve a ticket. However, this functionality does exist and it provides certain features unavailable for users who did not create an account, such as a list of all the past tickets they have submitted.

Even though inherently it is not a problem to create an account to access that functionality (and any other that can be implemented in the future), it is a layer of both complexity, hassle, and information that can be eliminated by implementing 3rd party authentication services, such as Google Sign-In, Facebook Connect, Twitter Connect, and other similar services.

This would completely eliminate the need of having to create (and therefore, remember) a new user for both creating tickets and using the system as a whole, which would also speed up the process. This will be implemented in a future release.

5.4 Observations and Conclusions

One of the most interesting findings is coming to terms with the realisation on how many people use mobile devices for all sort of tasks on a daily basis.

In its early stages, the application was meant to be fully mobile-friendly (and ended up being just part mobile-friendly) but due to time constraints, this was de-prioritised, as an assumption was made that it would not have a significant impact in the research.

Technically, this is true –not having a fully mobile version did not hinder the research nor skewed the results– but it did create an interesting side effect: mobile users did bring up that a mobile version was not available and they would like to have a better experience there,

even if it was good, demonstrating the importance of mobile-first design, and readily-available applications to be used in any sort of environment.

This would explain companies like Zendesk developing mobile applications, even if their main consumers are desktop users, as it is very important to cater to as many people as possible. Also, this reinforces one of the early design goals: when both the agent and the customer are available at the same time, they must be able to attend to their issues regardless of their location, which in today's world, it is equally likely to be on the run than in front of a desk.

Another important aspect that was unravelled with the results of this research was that customer satisfaction is not necessarily uniquely tied to the answer that was received (regardless of the outcome) nor the channel used, but the amount of steps taken to get to the same result.

Large scale companies like Google or Facebook rely on self-serving applications and knowledge bases that will try to resolve every single possible issue that a customer may have, but the more nuances exist, the less effective these become, and the necessity of a live support agent becomes more prevalent.

The increasing frustration that a customer builds up whilst browsing endless pages of articles without an answer and the realisation that the company is trying to obfuscate the path to get in touch with a support agent with hope of seizing a ticket deflection opportunity will decrease the satisfaction the customer in the end, regardless of how speedy or convenient the solution is after all this extra work was put.

This is supported in this research by the feedback provided by users, saying that the application was “minimalistic” and that one of the best aspects is its “simplicity”. This was intentional, as the author tried to follow the Unix Philosophy of “do one thing and do it right”. In this case, it was about being able to open tickets in a painless way, and getting a response the same way. By adding more and more layers between the agents and the customers, the experience becomes less ideal.

Companies have to be aware of this, although it is not as simple as eliminating all these layers all at once, since disregarding ticket deflection opportunities can be detrimental to productivity and costs. These findings suggest that it is mandatory to properly design self-service solutions that are as simple and straightforward as possible, without being labyrinthine or overly complex or explanatory on purpose.

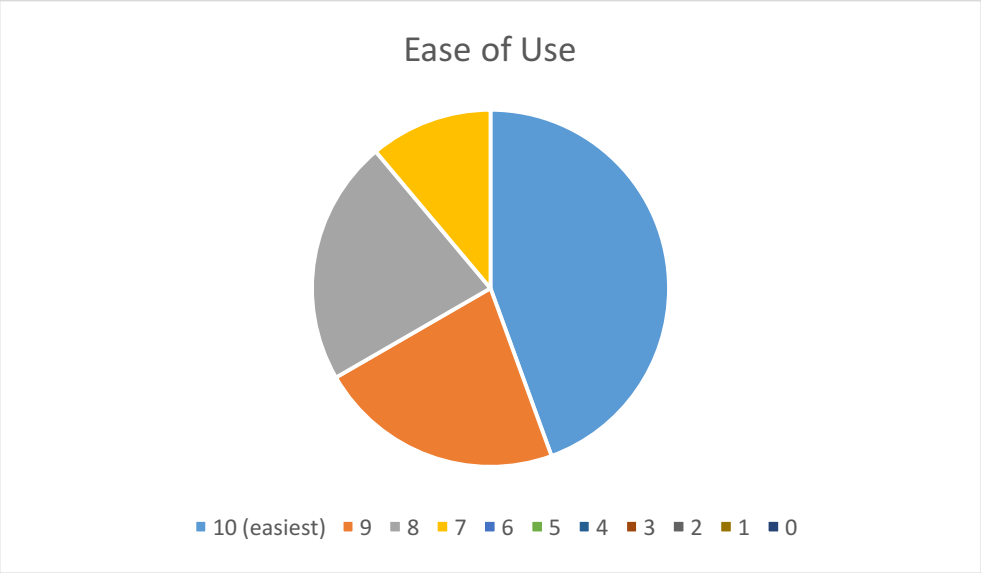

Fig. 4: User’s rating of the application’s ease of use. 10 is the highest in the scale.

As an extension of this, the general consensus is that the proof-of-concept was indeed very easy to use (scoring an average of 9/10 in

ease of use) and all subjects would like a similar experience in the future, using a tool like the one that was presented. Considering that every user had a different experience, like using the chat only, or the ticket system only, or a mixture of the two because of both the author's and the user's availability, this is a satisfactory result, as it proves that using both methods interchangeably and understanding that they both exist and readily available does cause a great and long-lasting, positive impression.

References

Gaur, A, & Arora, N (2014), "Effect of Age and Gender on Brand Loyalty and Customer Satisfaction-A Study of Mobile Phone User", *SIES Journal of Management*, 10, 2, pp. 22-30, Business Source Complete, EBSCOhost (Accessed: 12 April 2015).

http://search.ebscohost.com/login.aspx?direct=true&AuthType=ip,custuid,cookie,url,athens&custid=s6175963&db=bth&AN=99420274&site=eds-live

Stack Overflow (2015). "Stack Overflow Developer Survey 2015". (online) Available at: http://stackoverflow.com/research/developer-survey-2015#tech (Accessed: 17 Apr. 2015).

Schwartzman, R. (2013) "Reviving a Digital Dinosaur: Only Synchronous Online Chats and Peer Tutoring in Communication Centers" (Accessed: 7 July, 2015)

http://search.ebscohost.com/login.aspx?direct=true&AuthType=ip,custuid,cookie,url,athens&custid=s6175963&db=a9h&AN=93813995&site=eds-live

Forrester (2010), "Making Proactive Chat Work" (online) Available at: https://www.forrester.com/Making+Proactive+Chat+Work/fulltext/-/E-res57054 (Accessed: July 7, 2015)

Spolsky, J. (2000) "Painless Bug Tracking" (online) Available at: http://www.joelonsoftware.com/articles/fog0000000029.html (Accessed: July 9, 2015)

Zendesk (2015) "The Zendesk Benchmark Q1/2015" (online), Available at: https://d26a57ydsghvgx.cloudfront.net/content/resources/zendesk-benchmark-Q1-2015.pdf (Accessed: 31 Jul. 2015).

Schlesinger, R. (2014). *The 2015 U.S. and World Populations*. (online) US News & World Report. Available at: http://www.usnews.com/opinion/blogs/robert-schlesinger/2014/12/31/us-population-2015-320-million-and-world-population-72-billion (Accessed 3 Aug. 2015).

Statista, (2015). *United States: number of digital shoppers 2018 | Statistic*. (online) Available at: http://www.statista.com/statistics/183755/number-of-us-internet-shoppers-since-2009/ (Accessed 3 Aug. 2015).

Smith, C. (2015). *The surprising facts about who shops online and on mobile*. (online) Business Insider. Available at: http://uk.businessinsider.com/the-surprising-demographics-of-who-shops-online-and-on-mobile-2014-6?r=US&IR=T (Accessed 3 Aug. 2015).

RJMetrics, (2015). *2015 Ecommerce Growth Benchmark*. (online) Available at: https://rjmetrics.com/resources/reports/2015-ecommerce-growth-benchmark (Accessed 3 Aug. 2015).

Types of products purchased connected consumers in the past three months as of July 2014, c. (2015). *Online shopping categories in selected countries 2014 | Statistic*. (online) Statista. Available at: http://www.statista.com/statistics/410987/online-shopping-categories-countries/ (Accessed 3 Aug. 2015).

Buenger, M. (2015). *Shout-out: Are You Using Reddit for Public Health Research? What Works?* (online) Healthmediacollaboratory.org. Available at: http://www.healthmediacollaboratory.org/shout-out-are-you-using-reddit-for-public-health-research-what-works/ (Accessed 4 Aug. 2015).

Royal Holloway Library Blog, (2014). *Using Reddit for research*. (online) Available at: http://libraryblog.rhul.ac.uk/2014/08/18/using-reddit-research/ (Accessed 4 Aug. 2015).

Radigan, D. (2015). *A Brief Introduction to Kanban | The Agile Coach*. (online) Available at: https://www.atlassian.com/agile/kanban (Accessed 5 Aug. 2015).

Web App Frameworks (2015). *The Ruby Toolbox*. (online) Available at: https://www.ruby-toolbox.com/categories/web_app_frameworks (Accessed 6 Aug. 2015).

RailsGuides, (2015). *Active Record Migrations*. (online) Available at: http://guides.rubyonrails.org/active_record_migrations.html - active-record-and-referential-integrit y (Accessed 7 Aug. 2015).

Bertram, D. Voida, A. Greenberg, S. and Walker, R. (2010). “Communication, collaboration, and bugs: the social nature of issue tracking in small, collocated teams” (online) Available at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.148.2715&rep=rep1&type=pdf (Accessed 16 Aug. 2015).

Matsumoto, Y. (2008). “Ruby 1.9” (online). Available at: https://www.youtube.com/watch?v=oEkJvvGEtB4 (Accessed 16 Aug, 2015)