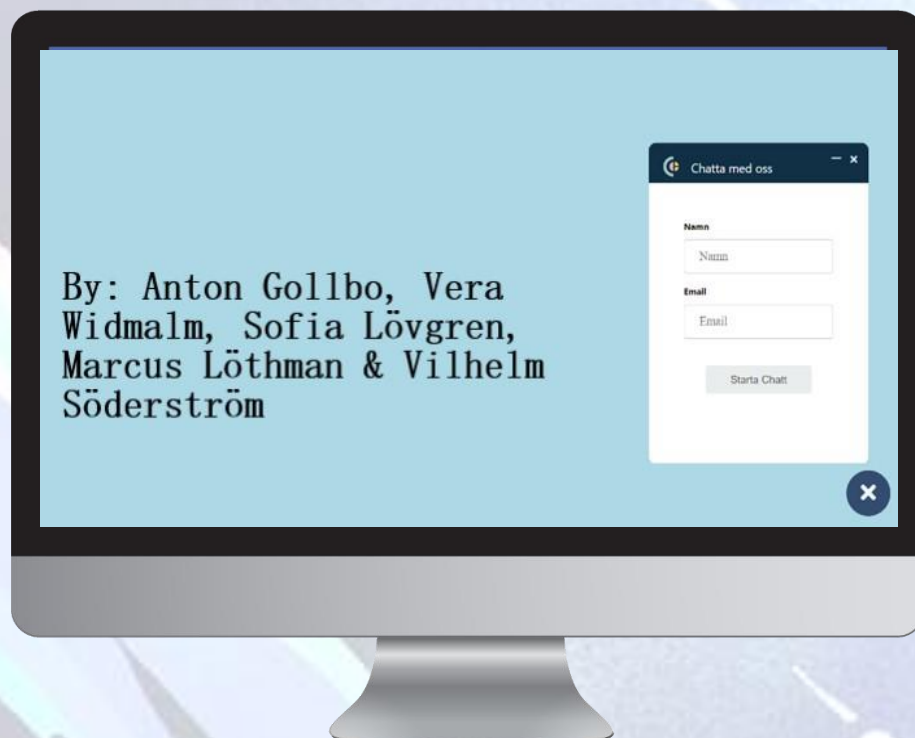


Scalable Chat Application and Chat Bot

- With Connectel



Supervised By:
Davide Vega Aurelio, Lecturer
Miran Nadir, Assistant
Jens Leijon, CTO at Connectel
Siavash Goudarzi, Full stack Developer at Connectel

Table of Contents

Abstract	3
1. Terminology.....	4
2. Introduction.....	5
2.1 <i>Connectel - Company introduction.....</i>	5
2.1.1 <i>Chat solution</i>	5
2.1.2 <i>Project goals.....</i>	5
2.1.3 <i>Scalability</i>	6
2.1.4 <i>Design</i>	6
2.1.5 <i>ChatBot</i>	6
3. Method.....	7
3.1 <i>Agile Development & Kanban</i>	7
3.2 <i>Team Structure.....</i>	9
3.3 <i>Collaboration and Communication</i>	10
3.3.1 <i>Collaboration with Connectel</i>	11
3.3.2 <i>Communication</i>	11
3.4 <i>Project plan</i>	11
3.4.1 <i>Project start-up</i>	11
3.4.2 <i>Project plan</i>	12
3.4.3 <i>Design.....</i>	12
3.4.4 <i>Functions.....</i>	12
3.4.5 <i>Usability.....</i>	12
3.4.6 <i>Thesis</i>	12
3.4.7 <i>Close Project</i>	12
4. User profiling	13
4.1 <i>Manager</i>	13
4.2 <i>User.....</i>	13
5. Technical solutions.....	15
5.1 <i>Frontend.....</i>	15
5.1.1 <i>JavaScript</i>	15
5.1.2 <i>CSS.....</i>	15
5.1.3 <i>HTML</i>	15
5.2 <i>Backend.....</i>	15
5.2.1 <i>MySQL</i>	15
5.2.2 <i>Node & Express</i>	16
5.2.3 <i>Axios.....</i>	16
5.3 <i>Management</i>	16
5.3.1 <i>Trello</i>	16
5.3.2 <i>GitHub</i>	17

6. Graphical User Interface- development	18
6.1 <i>Interface flow</i>	18
6.1.1 Chat	18
6.1.2 Management page	26
6.2 <i>Frontend design</i>	31
6.2.1 Chat	31
6.2.2 Chat - Usability	31
6.2.3 Chat - Color Scheme	32
6.3.4 Management	33
6.3.5 Management – Usability	33
6.3.6 Management – Color Scheme	34
7. Backend-development	35
7.1 <i>ChatBot</i>	35
7.1.2 Routes and answers	35
7.2 <i>Management</i>	37
7.2.1 Users	38
7.2.2 Management_actions	39
7.3 <i>Messaging through HTTP requests</i>	40
7.3.1 POST	40
7.3.2 GET	40
7.3.3 PUT	41
7.3.4 Polling and messaging	41
7.3.5 Sending of files	44
7.3.6 Token verification through GET and POST requests	45
7.3.7 Cookies	47
7.4 <i>Security</i>	49
7.4.1 Authentication	49
7.4.2 Security risks	51
8. User testing	52
9. Personal reflections	53
9.1 <i>Marcus Löthman</i>	53
9.2 <i>Vilhelm Söderström</i>	56
9.3 <i>Vera Widmalm</i>	59
9.4 <i>Anton Gollbo</i>	61
9.5 <i>Sofia Lövgren</i>	64

Abstract

Communication is key. Communication between human beings is one of the most basic functions for the world to function and mainly, to solve all the issues that might reside on our earth in our time. In marketing and supplying customers in any way, communication is paramount. In the world of online trade, a chat solution for customers is an accessible tool for customers to GET in contact and GET assistance. This thesis illustrates how a group of five STS students created a chat application for a customer service company called Connectel. The purpose of the thesis is to explain in general terms what choices were made and why in terms of technicalities and design. The purpose is further to illustrate how the application works and how that functionality was created. The requirement for the project was that it needed to contain a frontend and backend to create a web application or mobile application which could solve a problem at hand. This was done through division of the team into frontend and backend and collaborating toward the common goal. Roles were then further given within the divisions to better focus the individual skills within the team to smaller tasks. The end product that was produced is not a final product which is ready for the market. For this to be the case, the product requires further work.

1. Terminology

Chat – The term chat refers to the application for chatting and the chat-window itself.

Agent – The term agent refers to a real-life person working as a customer service agent which has the task of answering questions and helping customers.

Client – The term client refers to a customer / user wanting help with something and using the chat to seek assistance.

Management page – Page designed to configure and edit elements of the "ChatBot" decision tree.

Login – Page designed for managers to login to the management site and be able to edit the "ChatBot".

REST – Representational State Transfer." Architectural style for building large-scale networked applications." The aspects of REST are including: Data elements are retrieved through an interface," components communicate by transferring representations of resources through this interface rather than operating directly upon the resource itself", the requests are stateless(1).

API – Application Programming Interface. An API describes software, or libraries that uses standard functions. The API is a specification of how software can be used by another piece of software. An example is weather data available from SMHI that can be extracted and used by another piece of software outside SMHI.

CRUD – Create, Read, Update, Delete. Acronym for four basic functions of storage.

ChatBot – ChatBot refers to the decision tree and "AutoString" handler that is used in the Chat application when a client has not yet been connected an agent. Its purpose is to serve as a decision tree containing different categories and subcategories which in turn contain questions and answers that the customer might be looking for. The purpose of this is to minimize the help needed from a customer service agent.

AutoString – Refers to the string-handling that a client can use to search the database of categories, subcategories, questions, and answers and display it directly to the customer. The purpose of this is to provide a quick answer to a question that the customer might have.

2. Introduction

Over the last few years, the online shopping market has kept on growing in Europe, and Sweden is no exception. A study from 2016 showed that 21% of Swedish companies' sales were from the online market. This was an increase by 3% compared to a previous study in 2010. It is also 5% higher than the EU average of 16% (2). The wide market for selling products and services online puts a demand on clean and user-friendly websites. But although many sites provide convenient solutions considering returns, payment, shipping etc., visiting a site can be confusing for many users. A lot of sites therefore provide a support system in terms of a chat online. In a report from Svensk Handel, online-commerce manager Linda Hedström describes this as a crucial part of the business for providing a good working online marketplace (3). With this in mind, we can conclude that websites' customer support systems play a big role today, along with the online market growing. Another customer support solution that is getting more popular among companies is having a chat bot implemented in the support chat application. This provides the customer with some options on what their matter relates to. By choosing alternatives considering what the matter relates to, will result in less costs for the company, and can also save the customer some time.

2.1 Connectel - Company introduction

Connectel is a customer service software company from Uppsala. Their vision is to improve company- customer relationships by providing technical solutions (4)(5). They offer many types of modules within the so called "Connectel Solution", that is a cloud-based system with many types of features that work with the "omnichannel strategy". This is a strategy where all parts of the system are integrated and work with each other in a manner that appears seamless. With this way of working, you aim towards having a well-integrated system with the different subparts cooperating with one another. Some features in the solution include a webchat, social media, email, etc. For those to work as smooth as possible, Connectel offers tools like a scripting system, automatic callback and an automated FAQ generator based on the companies' most frequently asked questions over the phone (5).

2.1.1 Chat solution

Marketing manager Anna Itzel at Connectel writes about the importance of having a webchat at Connectel's website. She describes different factors that matter the most for the customer, for example fast response, and advantages of automating the chat with a ChatBot. Hence, a well working chat can make a company's business more lucrative. Not only does it save time in both company- and customer end, it gives an overall better impression of the website and the company (5). Therefore, Connectel provides a webchat module within their "Connectel Solution". The chat window is customizable, the customer gets to fill in initial information which simplifies for the answering agents, and the whole chat interaction is saved and monitored at the same place as the rest of the customer service modules.

2.1.2 Project goals

Connectel's webchat has some areas where there is room for development, according to CTO Jens Leijon. Meetings with him, and full stack developer Siavash Goudarzi lead the group to a few project goals.

2.1.3 Scalability

The existing chat is too "heavy" for the user traffic at different customer websites. Meaning, the code of the chat is located within a HTML element called iframe, that is loaded every time you visit the website, and the code itself is developed in a web framework called Angular. This leads to the chat loading very large files every time the window is used, which does not make it optimal in terms of scalability. This issue also has the consequence of the chat not being able to handle large amounts of network traffic. To solve this, the code of the chat would be written in plain JavaScript instead, avoiding any frameworks or libraries in the frontend of the chat. The code for the chat application should also not load until the chat window is opened.

2.1.4 Design

Another important feature of the chat is the chat window design. The pre-existing chat design is somewhat static, not so modern looking, and does not have a "clean enough" design according to CTO Jens Leijon. Although, the customer company that implements the chat has the ability to choose some color options in the manager site, but this needs to be simpler, with more "clean" and simple design options. Since more and more users visit the website via a phone nowadays, the chat should also be customized to a full screen-view when used by a mobile user. It is also important that first time users find the chat easy to use. This could also be achieved by making the chat as simple and user friendly as possible.

2.1.5 ChatBot

Alongside Jens Leijon and Siavash Goudarzi we decided that we also would aim toward building and adding a chat bot to the chat. This bot would be built by a decision-tree of options and answers to what the customer matter concerns. These questions, alternatives, and answers are then supposed to be modifiable by so called managers at the companies which implement the chat solution. Therefore, we also would need to make a manager login site where you could manage the bot. To make a management-site and a ChatBot we need to create a REST API that through HTTP-requests can insert and retrieve information in a database. Everything concerning the chat design would of course also be applied to the bot, i.e. a simple and plain design that is user friendly.

3. Method

The chat has been developed for 10 weeks and the team members have had various responsibilities during the process. This chapter we give an overview of which methods that has been used during the development, our project plan and team structure.

3.1 Agile Development & Kanban

The chat application has been developed with an Agile methodology which is an iterative, test-driven software development method. The method is based on a short feedback loop, where the planning is adapting to requirement changes during the process. Therefore, the project plan and time schedule are not set in stone. The team simultaneously works on different small parts of the project and to GET them function before implementing the whole system.

The Agile methodology also characterizes by its face to face communication and teamwork. Going into the project, the team decided that we should have working hours together, so it gets easier to collaborate and understand the different parts of the system. Each member started of the day by telling the group what they did yesterday and what they intended to do today. This prevented duplication of work and promoted cooperation, when for example two members realized they could implement their work or help someone out with something they have already developed.

There are a lot of different management methods that can be implemented to the agile methodology. We wanted something well-structured and straightforward and therefore chose the Kanban methodology. The Kanban board is an easy way to visualize the workflow by representing the work by its status. The different tasks get organized in columns which represents their status.

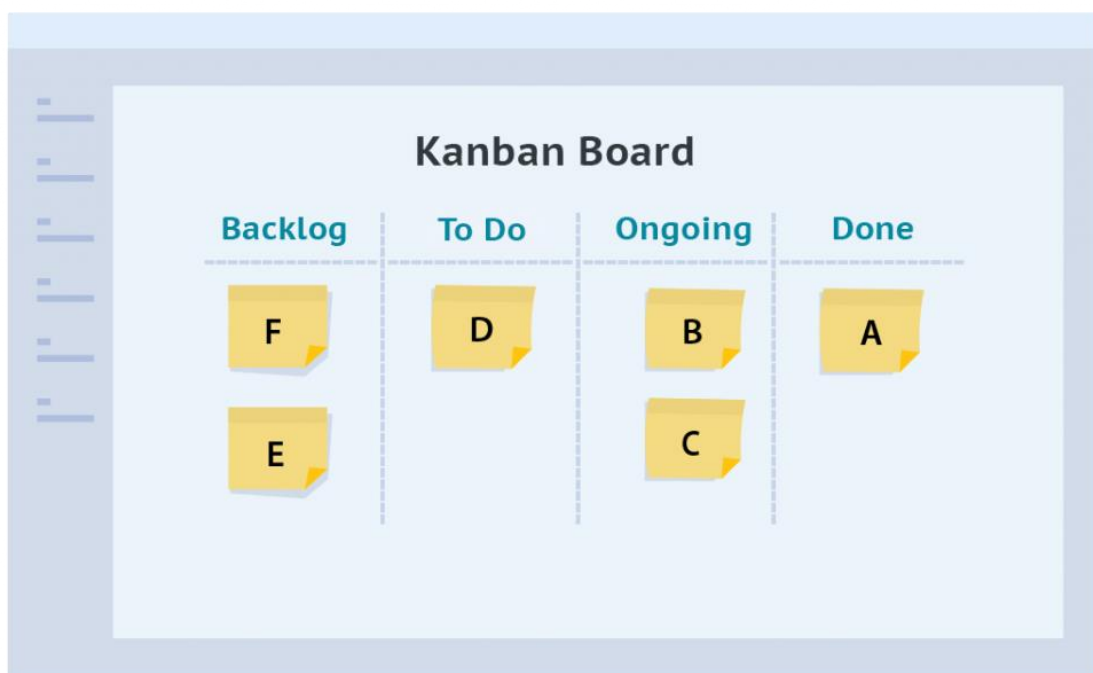


Figure 2.0: The Kanban Board (6)

We used Trello as the tool to set up our Kanban board. We also divided the “Done” column in different headlines, so we could easily follow our working process of a specific part of the project. The backend and frontend developers had separate boards that all members had access to. The Kanban board helped us keep track of our working process and increased our flexibility since it structured up the work and made it easier for the team members to move between tasks.

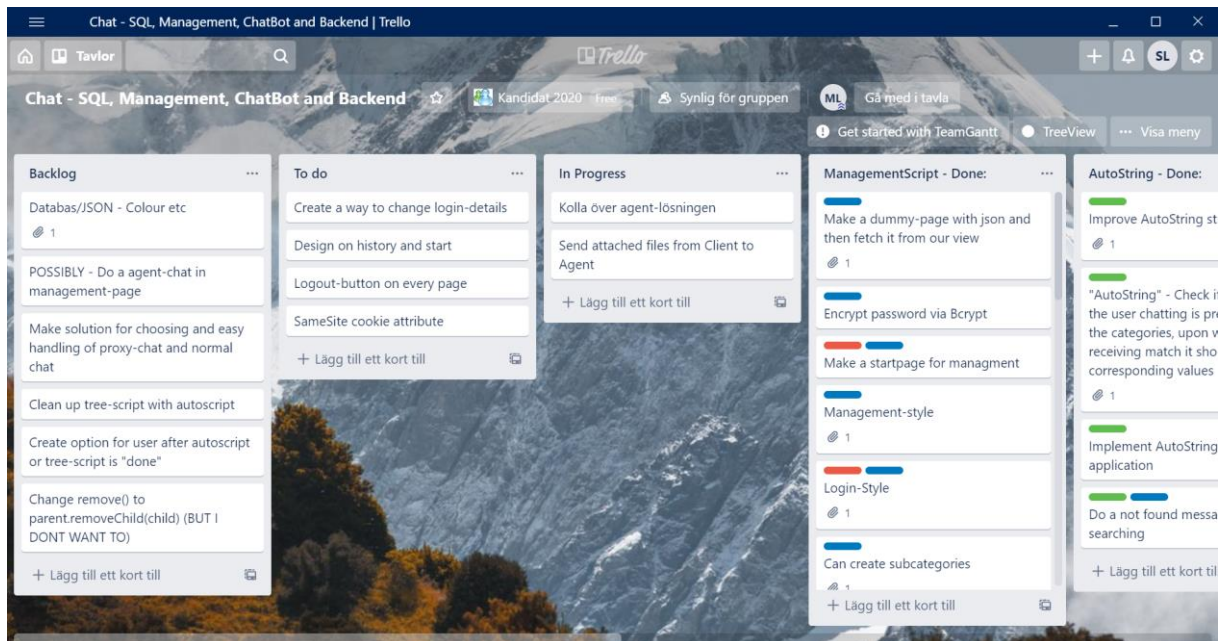


Figure 2.1: First part of the backend Trello board.

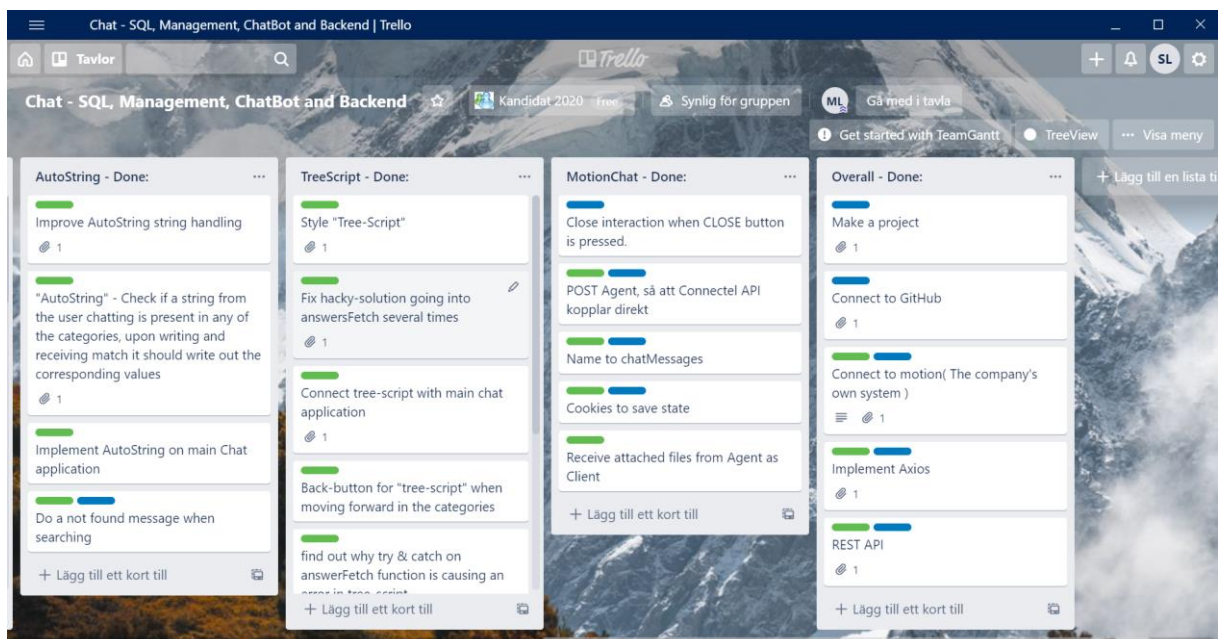


Figure 2.2: Second part of the backend Trello board.

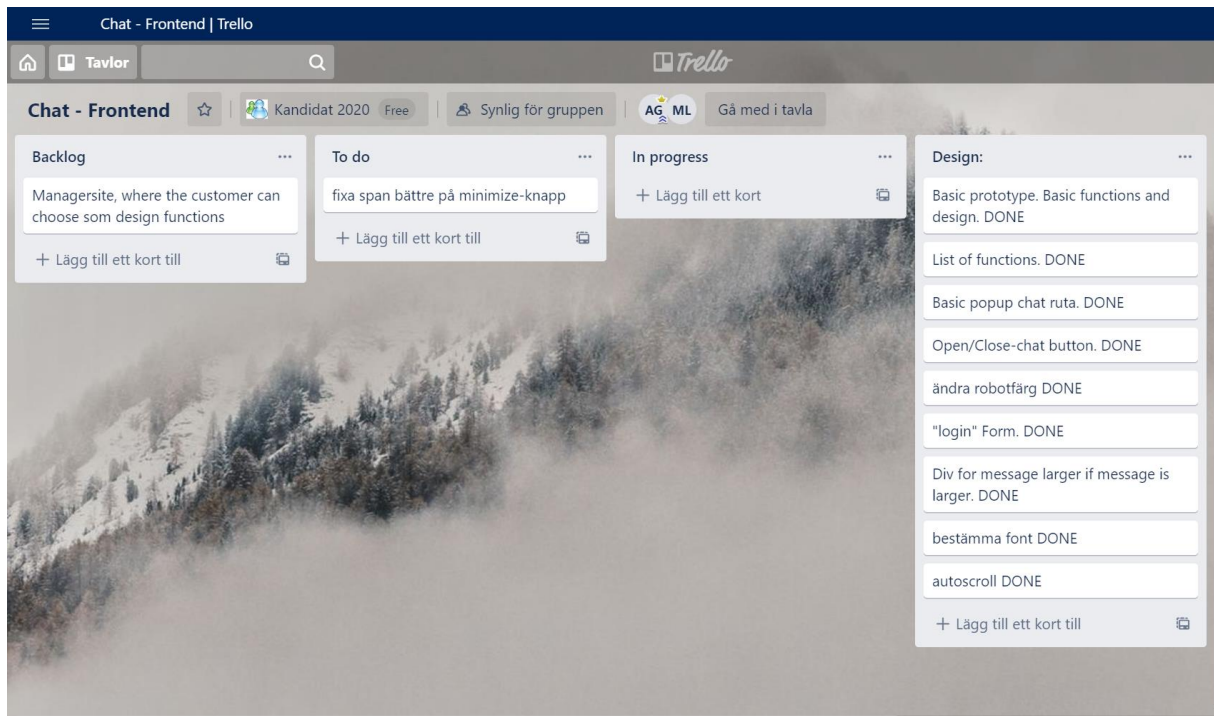


Figure 2.3: First part of the frontend Trello Board.

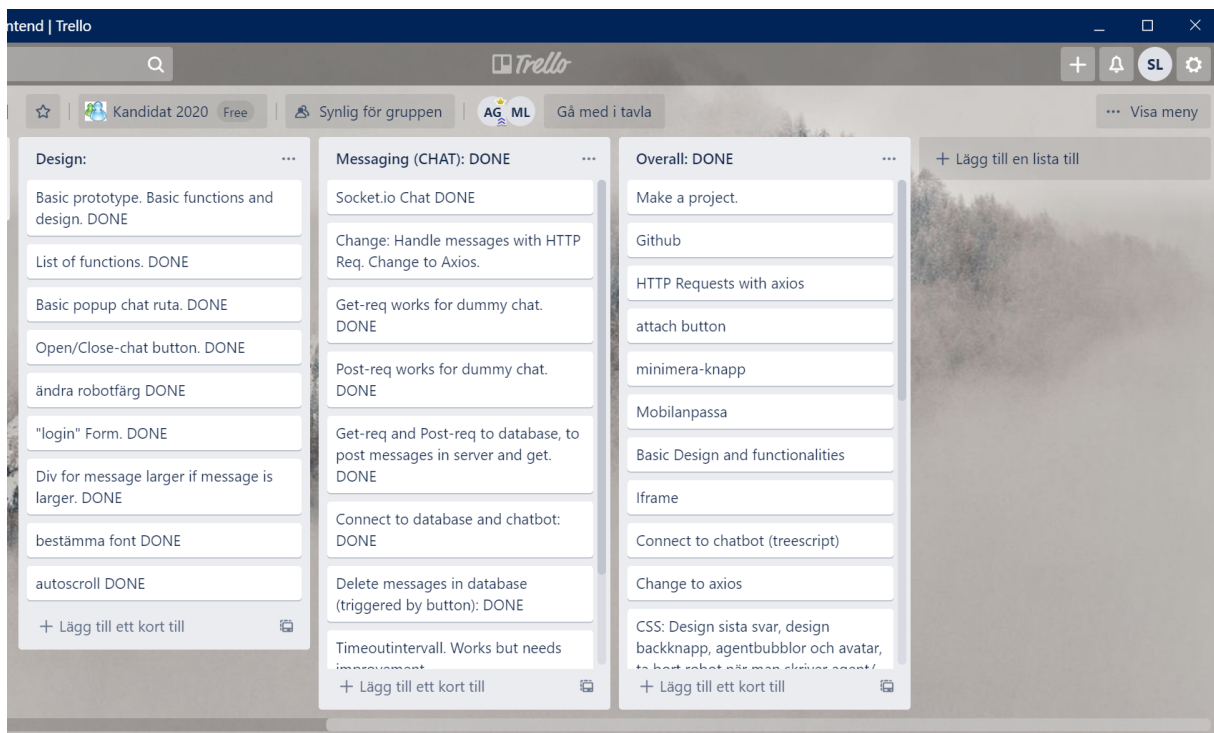


Figure 2.4: Second part of the frontend Trello Board.

3.2 Team Structure

The chat project is developed by five students, studying the Master Programme in Sociotechnical Systems Engineering. The project is a part of the Independent Project in Sociotechnical Systems Engineering focused on Information Technology systems.

The team is mainly divided in two groups: frond-end and backend developers. Apart from that, we decided to have a project manager and a communication manager. One provision going into the project, was that the group would meet up and have working hours together as much as possible. The purpose of this was to strengthen the cooperation, and to make it easier for every group member to have insight in every part of the project.

The roles were set based on each member preferences. Since the group had a good communication and collaboration between the frontend and backend developers, your role could sometimes change during the process.

Each role will be described below.

3.2.1 Project manager

The project managers main responsibility has been to have an overall view over the work and to make sure that all team members had responsibilities, and to guide the group in which tasks to prioritize when, to make the process more effective and smoother. The role was assigned to Sofia Lövgren.

3.2.2 Communication manager

The communication manager was the one who mainly kept contact with Connectel and with our supervisors at Uppsala University. He kept in touch with the developer and VD at Connectel, to let them know how the project went, and booked meetings if we had any questions. The communication manager was also in charge of the presentations and reports. Anton Gollbo were assigned to this role.

3.2.3 Frond-end developers

The three frontend developers of the team were responsible for the frontend programming of the project. They created the visual elements of the web-application, the features that are directly viewable and accessible by the client. This includes graphical design but also functionality. The project manager decided that the frontend developers also were responsible of making the messaging function of the chat to work. Sofia Lövgren, Vera Widmalm and Vilhelm Söderström.

3.2.4 Backend developers

The team had two backend developers, who were responsible for designing the database system and communication between database and browser. They worked mainly with the ChatBot and manager site.

3.3 Collaboration and Communication

The team had working hours every day, mostly between 8.30-16.00. As said, the day always started with a short meeting to keep everyone up to date what you were working on, and if some other task should be prioritized. Every Monday, the group had a meeting with our supervisor at Uppsala University where we showed our progress and got feedback.

Meeting up every day got us to collaborate more. We often solved problems together and at the end of the project, it was easy for us to implement the whole system since every member had insight in most parts of the project.

3.3.1 Collaboration with Connectel

Since we developed this project for Connectel, we had to keep contact with them to know what requirements to meet. The first two weeks of the project, we had around 3 meetings every week, to get all the necessary information to be able to start working. These meetings were with Jens Leijon, CTO, and Siavash Goudarzi, full stack developer. The meetings were held via ZOOM. We had a group chat with both Leijon and Goudarzi in Microsoft Teams throughout the project, where we could ask questions if we encountered a problem. About every two weeks, we had a ZOOM meeting with Goudarzi, to show him our progress and get feedback so we knew we were on the right track. At the end of the project we held a presentation for both Leijon and Goudarzi, where we showed them our work and went through the technical solutions of the web-application.

3.3.2 Communication

Communication between the group, outside of the working hours and in case of absence, was mainly held on Facebook's messenger. The team also had a Slack channel, with different conversations, one for the frontend development, one for the backend, and one general. The channel was mainly used for sending code or tips on useful videos and links. Slack was also used to keep in touch with our supervisors at the University. We got both general information and private feedback from them through slack.

Any lectures or presentations with our supervisors were held on ZOOM, due to the ongoing COVID-19 world pandemic.

To share documents with each other, we had a shared map on Google Drive, where we organized notes from meetings with Connectel or the University, presentation material etc.

3.4 Project plan

The Agile development methodology is characterized by its flexibility and iterative process, which makes the project plan constantly changing. Therefore, it was hard to structure the project from start since we knew that functions and requirements would be added and removed during the process. We made a GANTT-chart to keep a clear view of the work and a preliminary schedule, to not get stuck and spend too much time on one function.

The categories will be explained shortly below.

3.4.1 Project start-up

Includes decision making in setting up working conditions and assign roles. After deciding to collaborate with Connectel, we got introduced to the project and specified all the requirements and features of the web-application.

3.4.2 Project plan

Set up a preliminary project plan and schedule. Make a Trello Board.

3.4.3 Design

Make a clickable prototype and graphical profile. The building of the frontend framework and backend database will be done in parallel and integrated simultaneously during the process.

3.4.4 Functions

Includes the major and main functions and requirements for the project.

3.4.5 Usability

Testing the application user-friendliness, to make final touches.

3.4.6 Thesis

Start writing and dividing up the thesis.

3.4.7 Close Project

Make small adjustments and prepare for presentation to Connectel and the final presentation for our supervisors. Finalize the thesis.

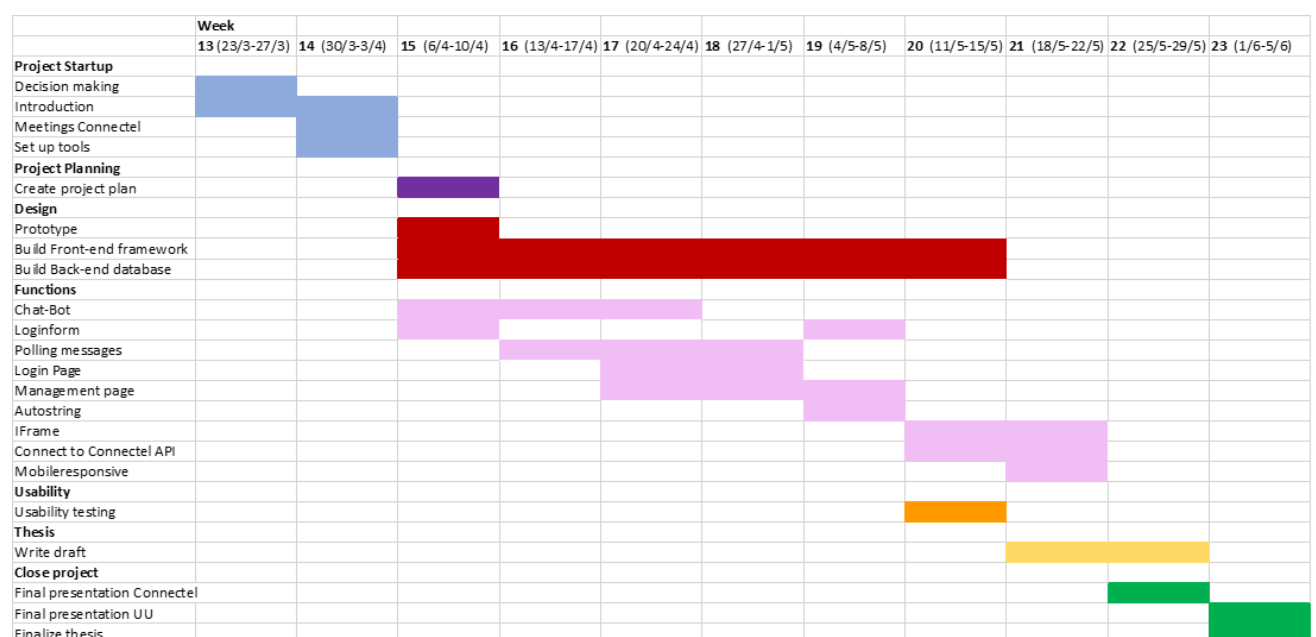


Figure 2.5: GANT-chart

4. User profiling

4.1 Manager

Name:

Monica Piedesh & Rackham Juventa

Occupation:

Monica: Founder and CTO of e-commerce website BuyStuff

Rackham: Customer service agent at BuyStuff

User objectives:

Monica is one of the founders and CTO of a medium-sized E-commerce company with limited customer service capabilities. Monica is also one of the managers of their customer service solution as she is the head of the technical solutions. They currently have one employee wholly dedicated to customer service, Rackham, and need to configure their solution accordingly. For this reason, they want to create a set up in the ChatBot that is as extensive as possible. Lars together with his customer service agent uses the Connectel solution to do this. Together, they will be able to keep track of what the commonly most answered questions are and react accordingly by providing answers to these directly in the ChatBot to reduce workload for Rackham. This will allow customer service agent Rackham to work proactively in other ways to help the company grow their customer service and work towards having a larger percentage of happy customers. As BuyStuff grows, the company's priorities need to be set and allowing for users to communicate with the ChatBot is an easy way to give Rackham the time and opportunity to dedicate his time to other more pressing matters at the company. Furthermore, Monica can together with Rackham continuously create an architecture for the ChatBot to be a blanket solution for most of the questions a customer might have.

4.2 User

Name:

Antoine Triesman

Occupation:

Insignificant

User objectives:

Antoine comes home one day to cook and realizes he needs to buy a new spatula. He searches the web for the best spatula provided in his price range and finds one he likes on BuyStuff's website. Antoine orders the spatula. He later finds he has a few questions regarding the spatula and visits BuyStuff's website yet again to find the answers he is looking for. He wonders what the estimated delivery time for the spatula is and decides to use the chat. He is faced with the initial ChatBot which tries to help him answer his questions directly. Using the AutoString feature of the ChatBot, Antoine searches for "Delivery time" and finds the answers directly. Now Antoine can wait in the comfort of his home for the delivery of the spatula.

Name:

Krister Sjögren

Occupation:

Singer

User objectives:

Krister is not happy. Krister has ordered a new microphone from the BuyStuff website. After two weeks he finds that the treble on the microphone has lost its original sound and wants to investigate what right he has to return the product or make a claim against the quality of the product. He decides to use the chat on the BuyStuff website to try and acquire information on what rights he has regarding this. On the website Krister cannot find what he searches for in the ChatBot and wants to connect to an agent instead. He writes "agent" in the chat window and after a short while gets connected to an agent. He asks the agent what right he has to return or file a complaint against the product and the agent helps him in the best way he can. Krister now knows he has the right to file a claim against the quality of the product and gets sent the right document to do so by the agent. Krister is happy again.

5. Technical solutions

5.1 Frontend

Frontend refers to the presentation or user interface of a web-application.

5.1.1 JavaScript

JavaScript is a “high-level” programming language, meaning it does not require and use direct access to computer architecture such as the CPU. This is because JavaScript was created for browser-based solutions. JavaScript is there used to create a functioning and interactive web environment, manipulating elements, and reacting to user input are two examples of this. Most of the code written in this project is done through JavaScript because it is supported by all major browsers and has full integration with CSS and HTML. (8)

The decision to use JavaScript was made because it was of the project description from Connectel, as our application had to be as lightweight as possible and not use any frameworks for JavaScript which would make it “heavier” (i.e. has to load data consisting of external elements) to run. In the beginning we found it hard to work with JavaScript but the fact the we worked without the use frameworks motivated us and as the time went we came to really understanding it and appreciate the knowledge we gained.

5.1.2 CSS

CSS is used to describe and configurate how HTML elements are being shown on screen. It is used to separate between functions and design of a web application. (9) CSS was in our project used for layout and styling of all the elements within the chat and management interfaces. It is the only tool used for design and styling of the content on the page since the use of frameworks was not an alternative, as the application had to be as lightweight as possible. To use CSS meant a lot of hours extra were spent on design compared if we used a framework.

5.1.3 HTML

HTML is “the standard mark-up language” for building website elements. Together with CSS and JavaScript it makes up the basic fundamentality of most websites today. (10) In our project, HTML elements were created within JavaScript to form the web application in its entirety. The decision to have the creation of most of the HTML-code in JavaScript was to enable that it could be transferred as a product in form of a code snippet.

5.2 Backend

Backend refers to the handling of data within the web-application.

5.2.1 MySQL

MySQL is a relational database management system. It is used for modelling and using relational databases.(11) In our project, the main uses of MySQL was the MySQL server and

MySQL workbench which is the UI for handling databases. MySQL server was used to setup a local server which together with Node and Express could communicate and work as a "mock" REST API.

The decision to use MySQL came out of two reasons. Firstly, we have previous experience with MySQL which we found easily maneuvered but secondly, and most importantly, it was also the recommendation from Connectel that considered MySQL to be the database framework they were going to work with in the future. The decision to use MySQL we believe helped us in the creation since it has a lot of useful functions that solved a lot of problems for us which would have been hard to come around without the functions. Although MySQL feels very robust and was easy for us to use, it would be nice to have a built in feature in which you could set up a local server easily for all of the group to connect to (which is probably possible, but not something we looked heavily into).

5.2.2 Node & Express

Node JS is an event-driven JavaScript runtime that is designed to build scalable network applications. It uses the Chrome V8 JavaScript engine to translate JavaScript to machine code. "Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications" (12). Express is used for building web applications and APIs. It is often used as the regular server framework for Node.js.

The decision here was made on our part because we used it extensively in a previous project and it was overall a good decision. Node.js has a lot of modules that helped us with aspects like encryption and authentication and worked smoothly for us throughout the entire project.

5.2.3 Axios

Axios is a JavaScript library used to ease the use of HTTP- and XMLHttpRequests. (13) Through Node it is used for CRUD operations to and from a REST API solution. In our project, Axios was used for database insertion through our web application and simplifying the use of HTTP-requests to Connectel's API.

Axios is a "promise-based" HTTP client system used for browsers and in Node. Axios is used to send asynchronous HTTP requests using REST APIs through CRUD-operations. Axios was recommended to us for the handling of HTTP- and XMLHttpRequests by Siavash Goudarzi at Connectel. Axios we found to be an excellent library that really helped us get through the last stages of development. It took some time to learn but when we had figured it out, we could accomplish the last unfulfilled parts of our project goals such as connecting our chat the Connectel's agent site

5.3 Management

Management refers to all the tools used for sharing code and planning the project.

5.3.1 Trello

Trello is a tool for organizing projects by dividing them into boards. Trello uses boards for organizing and handling of projects. In our project, we used a basic Kanban structure for organizing our work. Trello was then used to organize the work further within the boards and boundaries for different tasks which we created. We used Trello because we all had heard great recommendations of it and throughout the project it lived up to our expectations.

5.3.2 GitHub

GitHub is a developer platform used for sharing and saving code. (14) In our project GitHub was essential to share code amongst one another and keep track of versions and new updates. This was also an easy decision to use GitHub because of our own previous experiences in using it and our experience with it during this project worked just as we expected.

6. Graphical User Interface- development

In this chapter we go through the interface of the chat and management page from the user's perspective. Some essential functions concerning user friendliness, lightweight structure and clean design is explained, and commented on why they matter.

6.1 Interface flow

To make the product simple to use, a key part is the interface flow. By this, we mean that the user must feel that the order of the different steps is "logical". For example, when starting a chat process, it is common that the login state comes first, since the user is used to this being the initial step of chatting. Therefore, we go through the different "steps" of the pages to show what the user is experiencing when using our product.

6.1.1 Chat

When the user visits a website, and wants customer support, they can find the chat icon in the top right corner of the site. In the picture below, you can see the blue background representing any website window, with our chat button that will trigger a chat window if pressed.

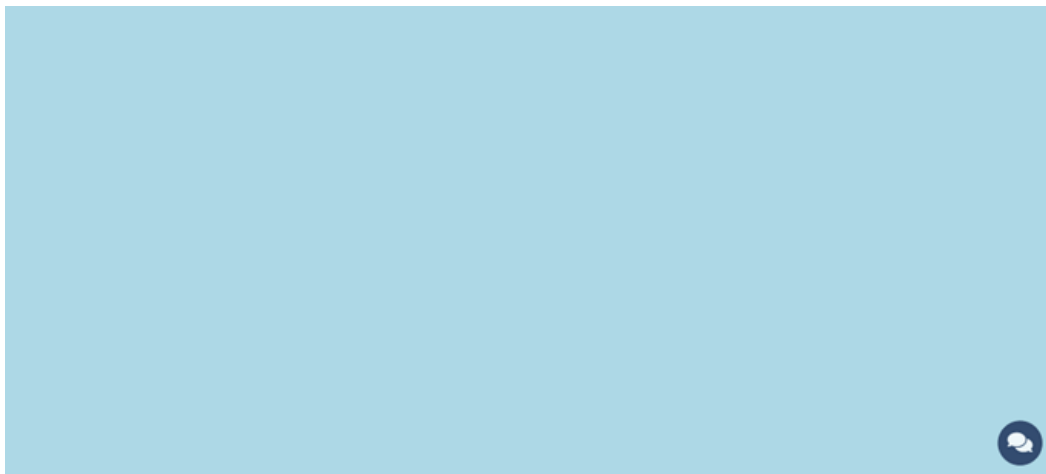


Figure 6.1: The first website view

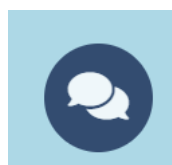


Figure 6.2: A zoomed in picture of the chat button

If you click this button, the chat will pop up just above the button, a little to the left which you can see in figure 6.3. The symbol with the chat bubbles will also change to a cross, representing that if you press the same button again, it will instead minimize the window down again.



Figure 6.3: Window view when the chat has popped up

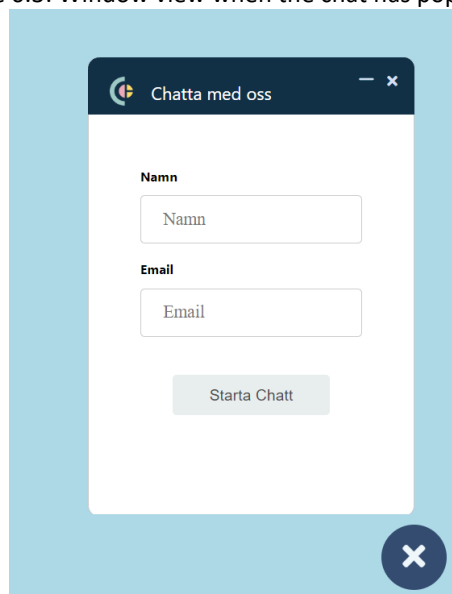


Figure 6.4: A zoomed in picture of the chat window

The first view the user gets is the login, just as described above. There is no possibility to do anything else than enter your information before getting to chat, which helps the step-by step- view we are trying to accomplish. After entering the name, with a valid email, the user can easily find the “Starta chatt” - button to start the chatting process. When this is done, the user enters the ChatBot- stage as you can see in the figure below.

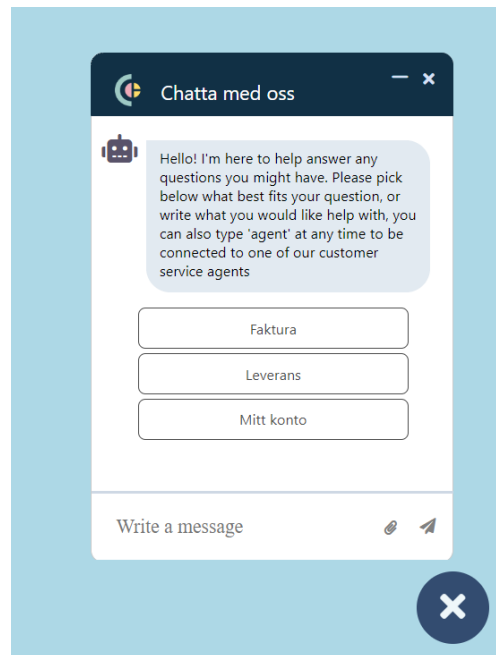


Figure 6.5: Getting to specify your matter with the chat bot

In this stage, you get to specify what your complication or question comprises via the chat bot. It welcomes the user by saying hello and describes how you can get its' help. Below, the alternatives of categories are displayed, from which the user's question might fit into. By choosing one of them, a subcategory, or sub- question might be displayed. By choosing a new category you, the complication can be specified further. This is done until the decision tree reaches its' end. The following example shows how the user navigates in the tree to get information about where to find the invoice number.

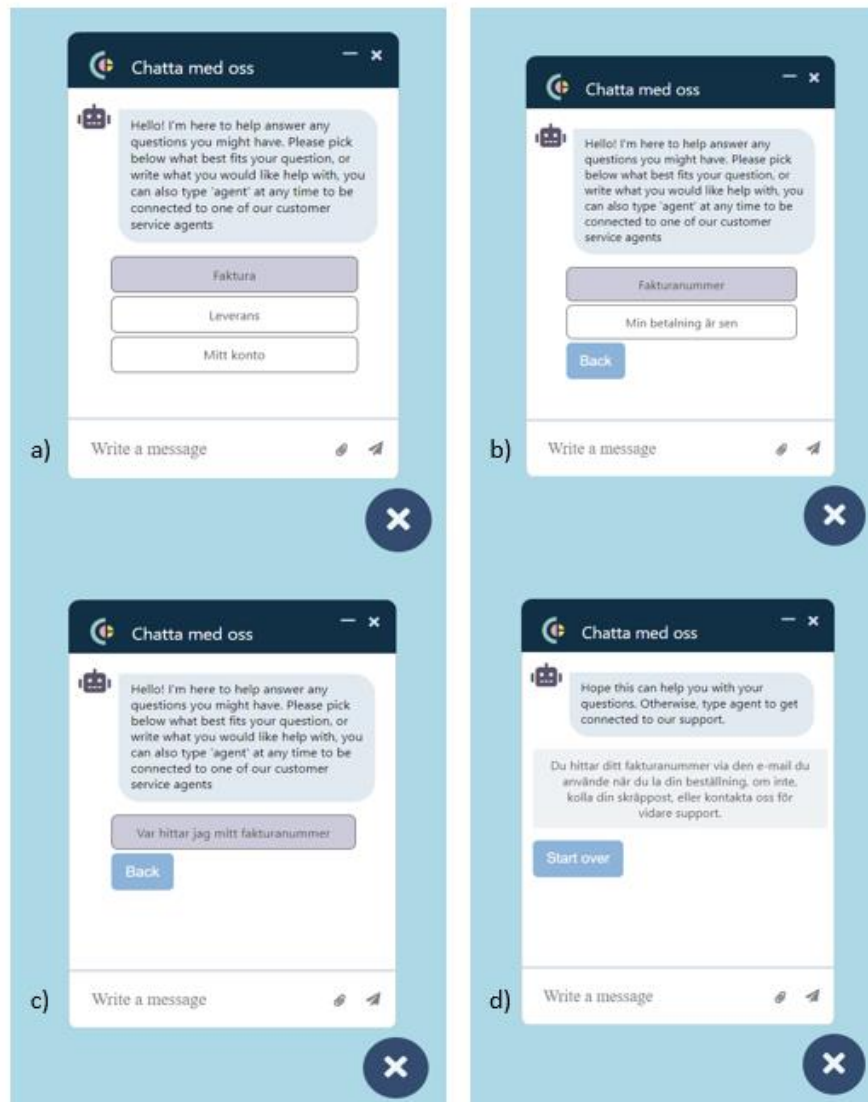


Figure 6.6 ChatBot Navigation

The user wants to know where to find what invoice number he or she got when ordering from the site last week. The ChatBot is used to find this out.

The steps:

- The user chooses between the options *invoice*, *shipping*, or *my account*, and identifies *invoice* to be the most suitable concerning his or her question.
- He or she finds the option *invoice number* which seems to be accurate, the user chooses that alternative.
- He or she finds the question *where do I find my invoice number* which seems to be accurate, the user chooses that alternative.
- The answer of that question is displayed. The decision tree has now reached its' end, and the bot says that you can type in "agent" to get connected to an agent to get further support. The user can also choose "start over" in order to start from the beginning to find answers to other questions.

If the user instead does not find any matching option, there is a possibility to search for any category. This process is displayed in the picture below.

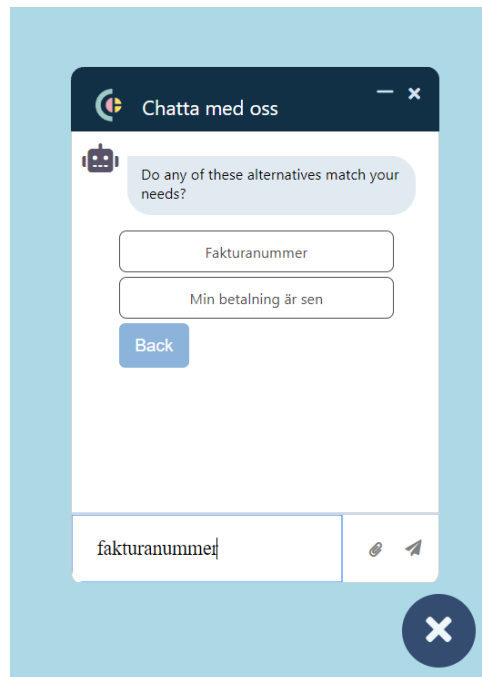


Figure 6.7: The user searches for anything regarding invoice number.

If the user is not pleased with the help from the bot, he or she can write “agent” to get connected with the human help support. An illustration of this is seen below.

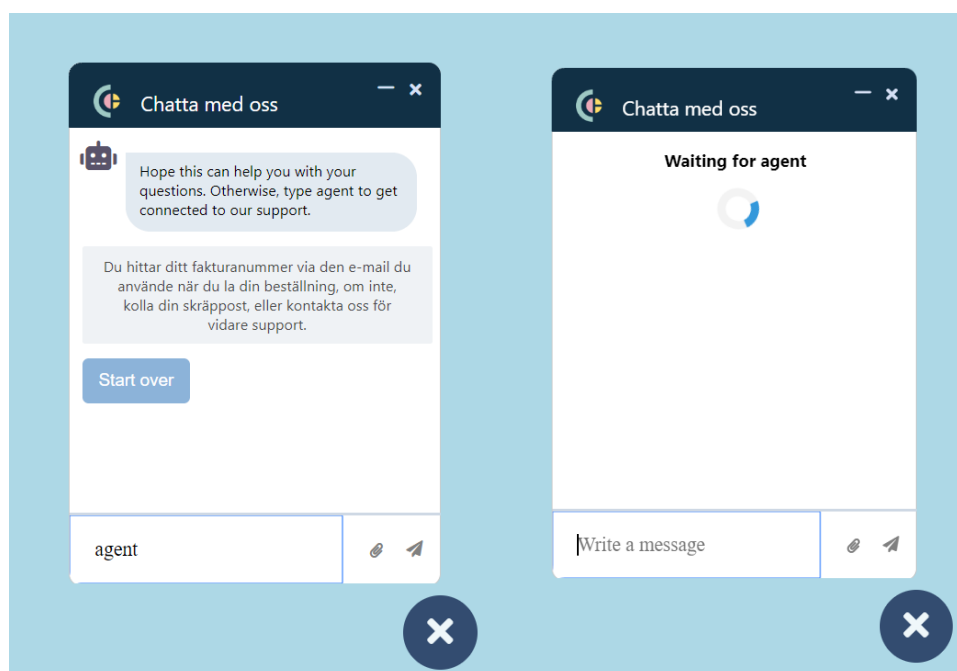


Figure 6.8 The user types in “agent” to get connected to the support chat.

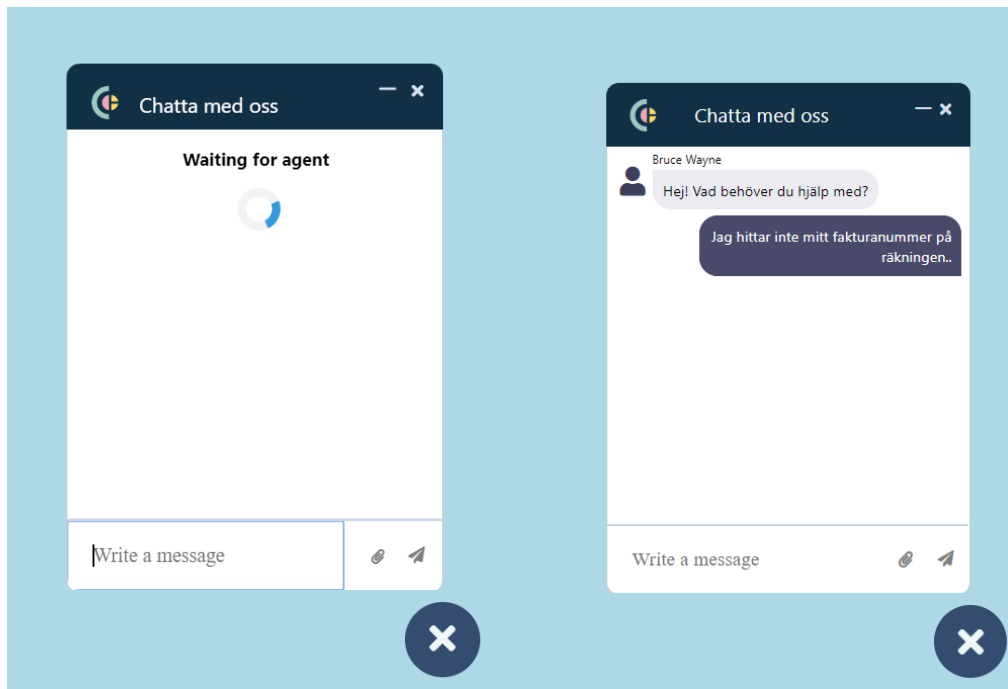


Figure 6.9: Waiting and then getting to chat with an agent connected with an agent

One feature we thought was useful was being able to attach files and sending them to the support, and vice versa. Therefore, the attach button with a symbolizing paperclip was implemented in the input window beside the send button.

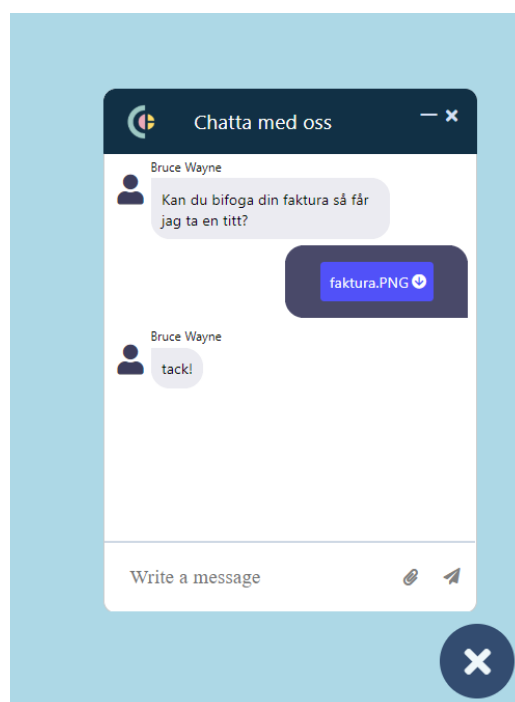


Figure 6.10: Attaching files via the attach button

When the user presses the close button in the top right corner of the chat window, the chat is supposed to close, i.e. the connection between the user and the agent closes. But first, a confirmation box is shown which is seen in figure 6.11. This is supposed to prevent mistakenly closing the chat, since the user might mix it up with the minimizing button.

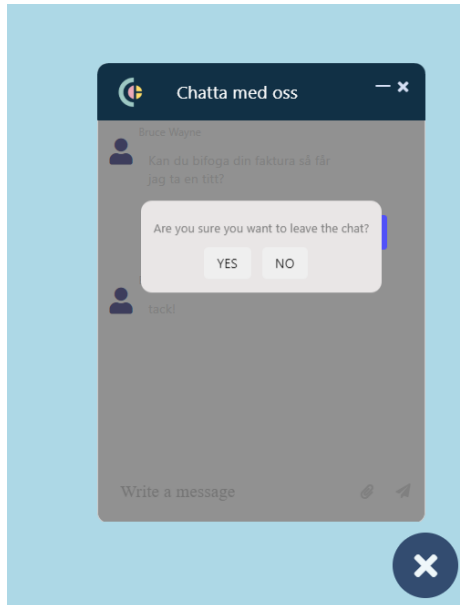


Figure 6.11: The user wants to close the chat but is asked if he or she is sure first.

If the user presses “YES” the connection and the whole window is closed. The round button’s cross, is replaced by the chat bubbles that was displayed in the beginning.

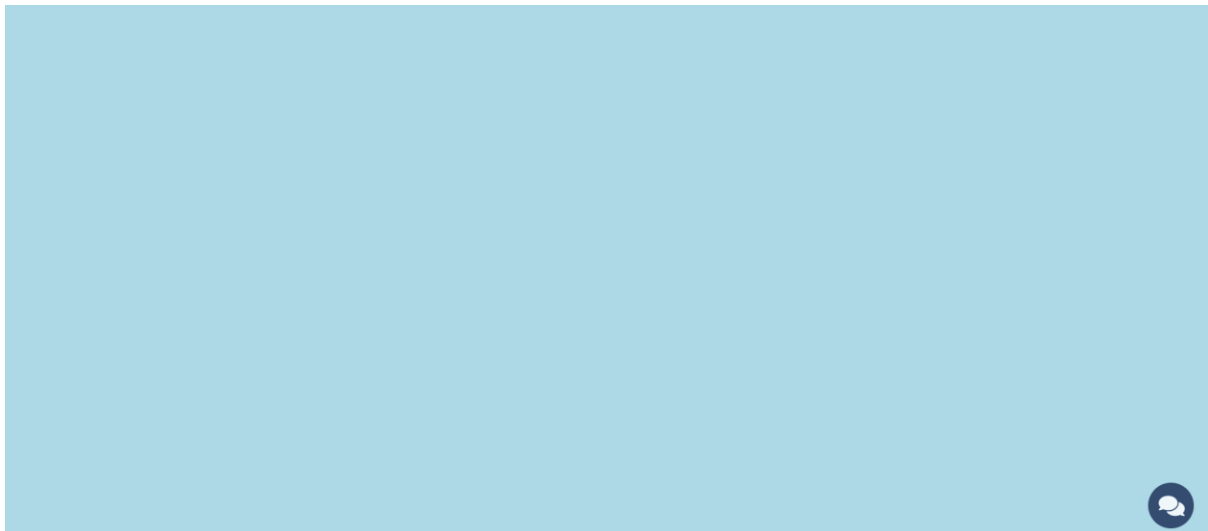


Figure 6.12 The user has closed the chat.

If the chat instead is closed by the agent on the answering side, the user gets a small message that the chat has been disconnected which is seen in figure 6.13.

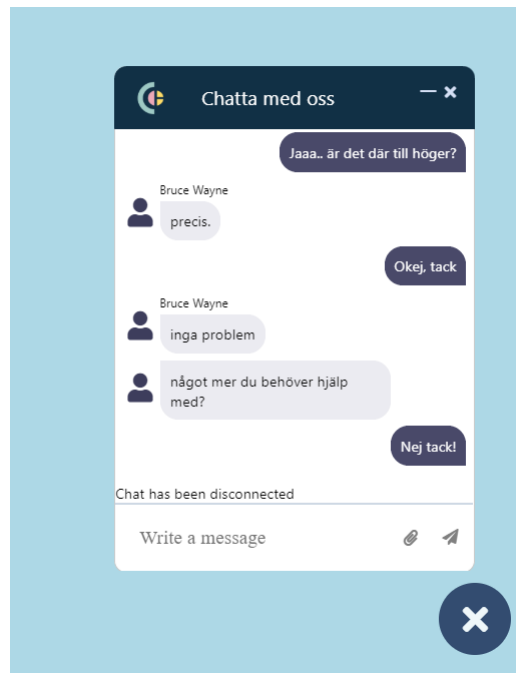


Figure 6.13: The answering agent has closed the chat.

As mentioned before, one key goal was to make the chat user friendly for mobile users. When discussing this with Siavash Goudarzi, we decided to make the chat covering the whole screen when used from on a mobile device. This was done by changing some positions to “absolute” in CSS and making most of the widths and heights to percentage instead of pixels in the CSS style document. The figure below shows how the chat would be displayed with a Galaxy S5 phone.

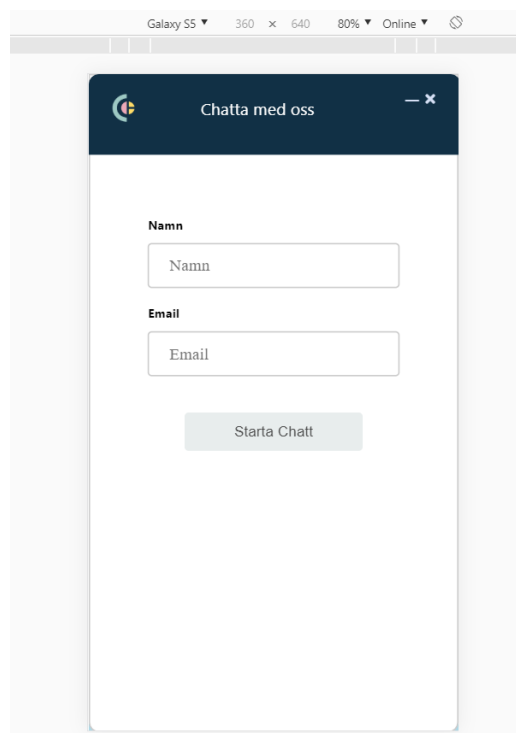


Figure 6.14: Chatting from a mobile device.

One last feature that was implemented, but hard to display with a picture, is the scroll behaviour of the output window, which is the middle part of the window between the blue panel and the input field. Code wise, we structured this by setting a scroll function that scrolls down the length of the total output window when a new message is posted. By setting the scroll-behaviour to “smooth” in the CSS file, we got a smoother feel to the scrolling. And since the scrolling only is done when a new message is posted, the user is still free to scroll by his or herself to look at earlier messages. The chat is not scrolled to the bottom all the time.

6.1.2 Management page

When a manager wants to visit the management sites the first thing, he/she will encounter is a log in page to gain access to the sites. This page is seen in the figure below

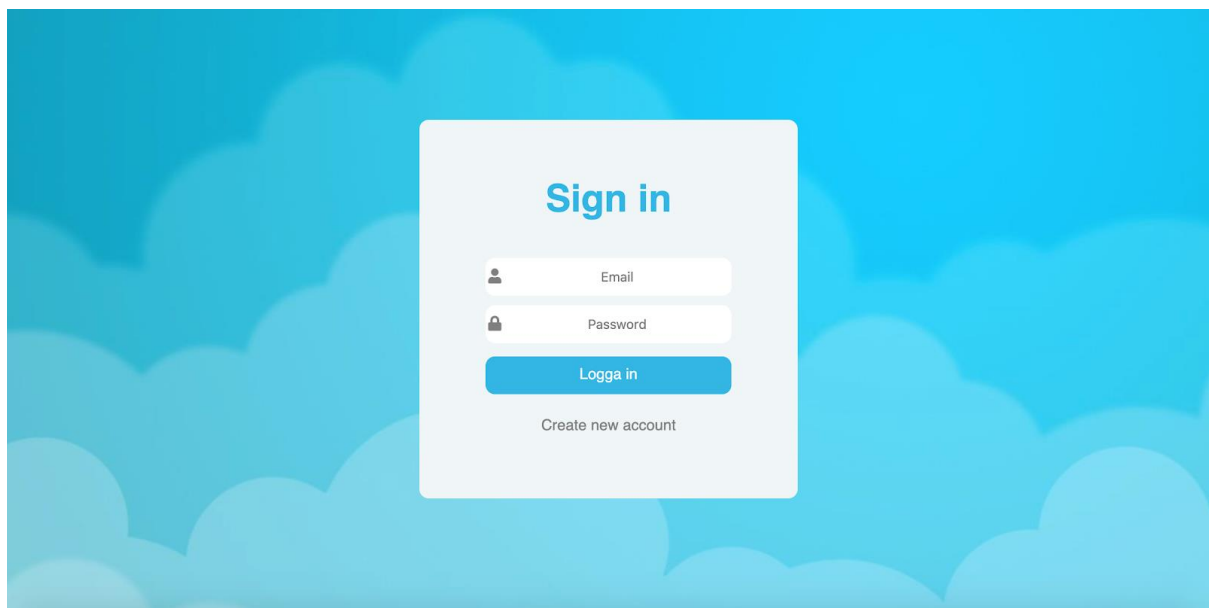


Figure 6.15: Login to the management page

If it is the first time the manager visits the page, he/she will need to create a new account. This is done by clicking the link “Create a new account” which will navigate you to a create new account page.

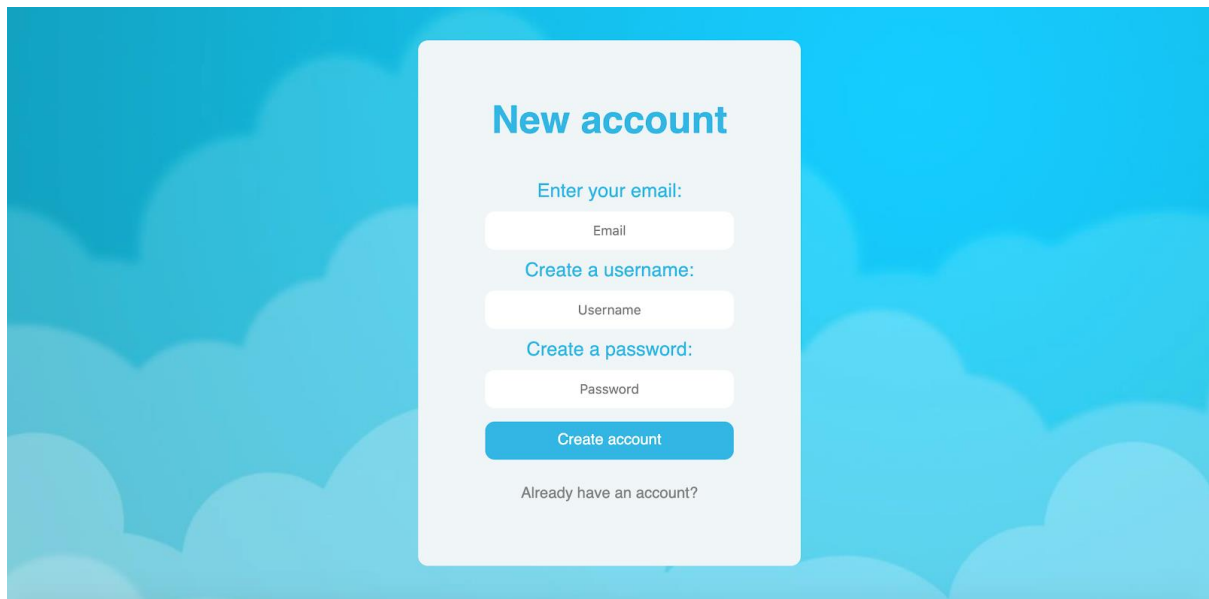
A screenshot of a 'New account' registration form. The form is centered on a light blue background with a cloud pattern. It has a title 'New account' in bold blue text. Below the title are three input fields: 'Email', 'Username', and 'Password', each preceded by a blue label. A blue 'Create account' button is at the bottom of the form, followed by a link 'Already have an account?' in a smaller font.

Figure 6.16: Creating a new account in management page

Here the manager needs to type in three fields: one for email, one for creating a username and one for creating a password. When pressing create account a POST method will request the filled in details from the body of the template and store them in variables. These variables values are then inserted into a user table in MySQL. If all the fields are filled in the user will be redirected to the log in page, if not the user will be notified that all the fields have not been completed which is seen in figure 6.16. There is also a function that validates the text in the email-field.

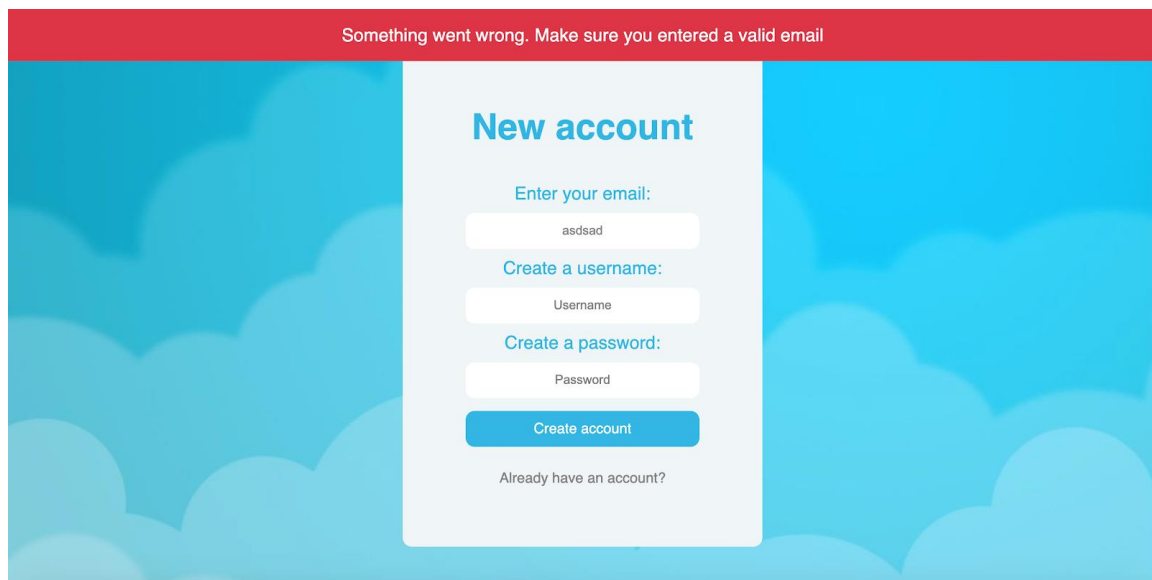
A screenshot of the 'New account' form with an error message. A red banner at the top of the form area contains the text 'Something went wrong. Make sure you entered a valid email'. The form fields are the same as in Figure 6.16, but the 'Email' field now contains the text 'asdsad'. The 'Create account' button and the 'Already have an account?' link are still present at the bottom.

Figure 6.17: Login to the management page

Once logged in, the user will be met by the page seen in figure 6.17 with a welcoming text including the username. The user will also have four clickable options. He/she can either choose to log out and then be redirected to the log in page. The user can also choose to navigate via buttons to the different management pages or open a navigation bar, in figure

6.18, that slides in from the left, also offering navigation possibilities and is available on all the pages.

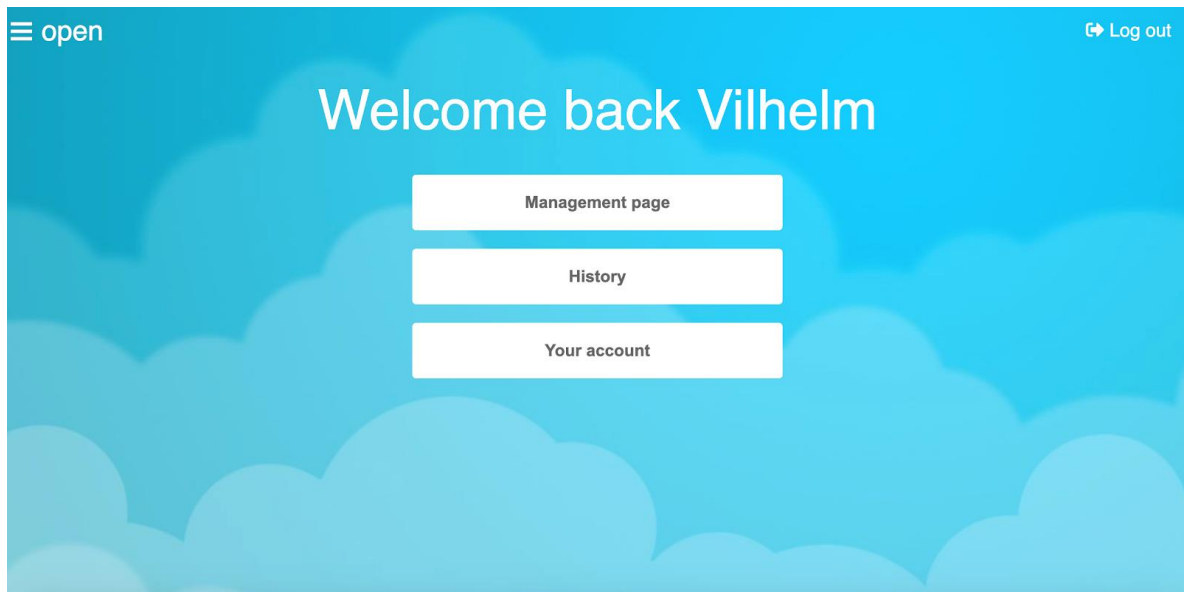


Figure 6.18: First page after login

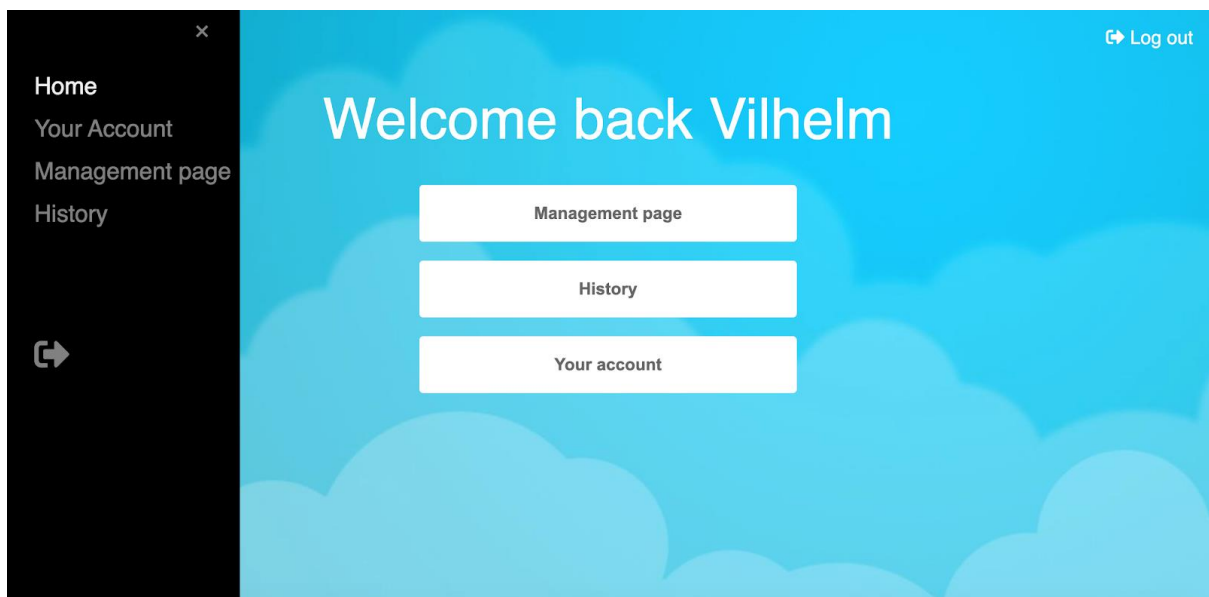


Figure 6.19: Sidebar

The management page in figure 6.19 is where the ChatBot can be managed and the most important page. Here the user can add, change, and delete the name of main categories. The user also has the option to manage main categories.

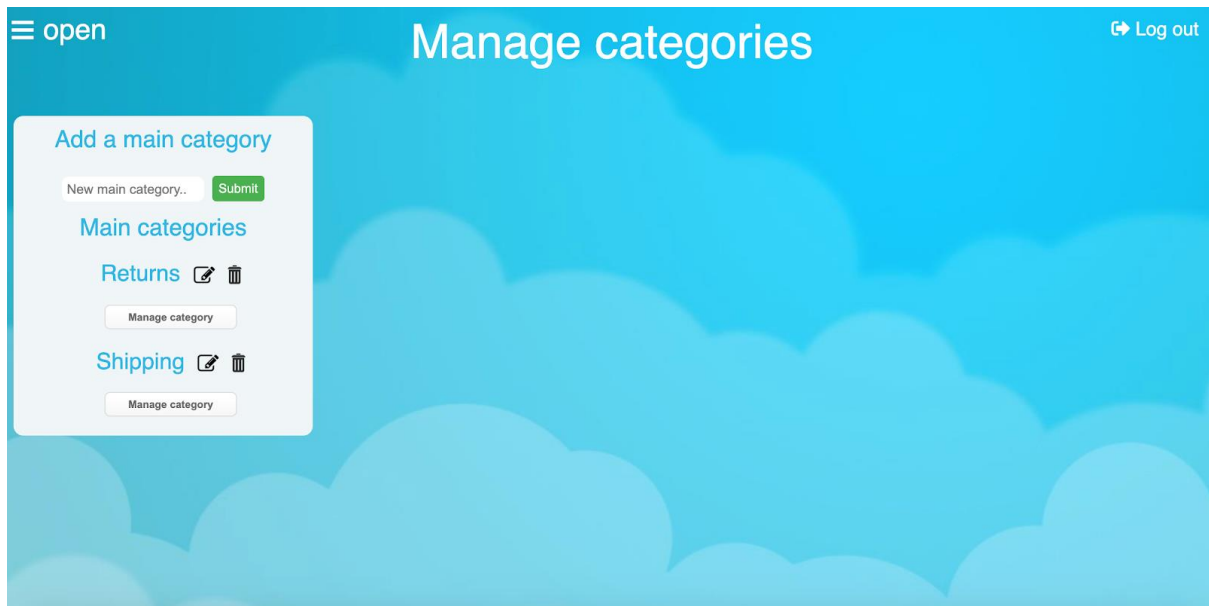


Figure 6.20: Manage categories page

When managing a category as in figure 6.20, the user can add questions and answers to that specific category. Users also have the option to add subcategories to the category. To each subcategory it is also possible to add new subcategories and questions in the same way as with the main categories.

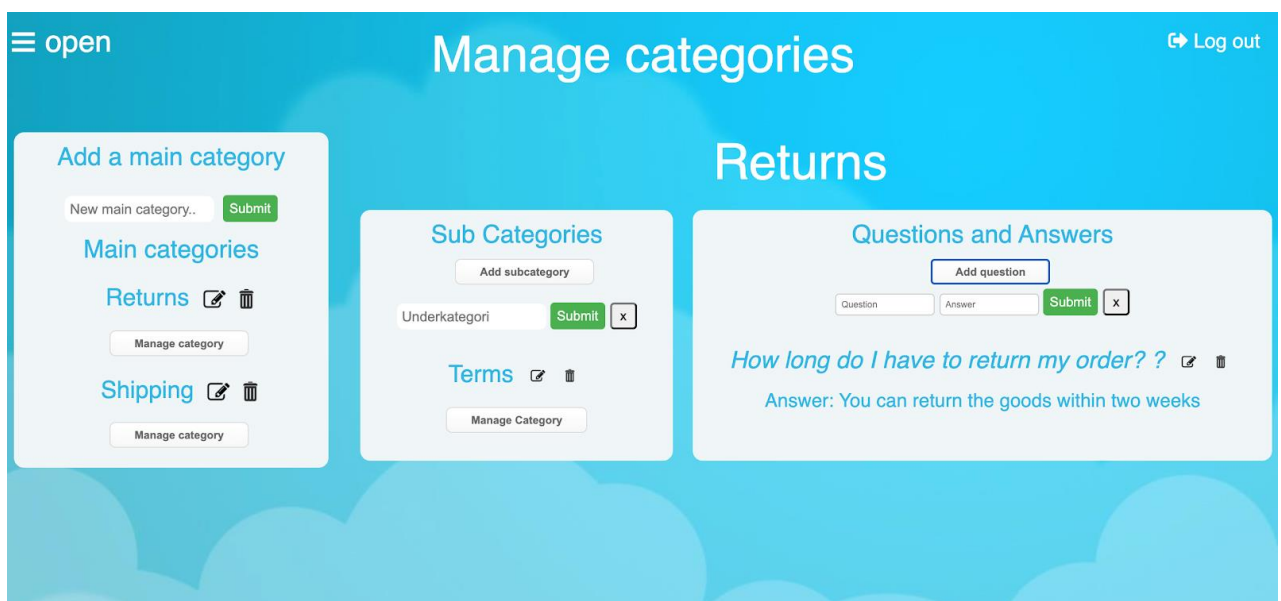


Figure 6.21: Adding subcategories and questions

On the History page in figure 6.21, the user can see all the changes made with the ChatBot from the manage categories page in a scrollable “list” and which user that made the changes.

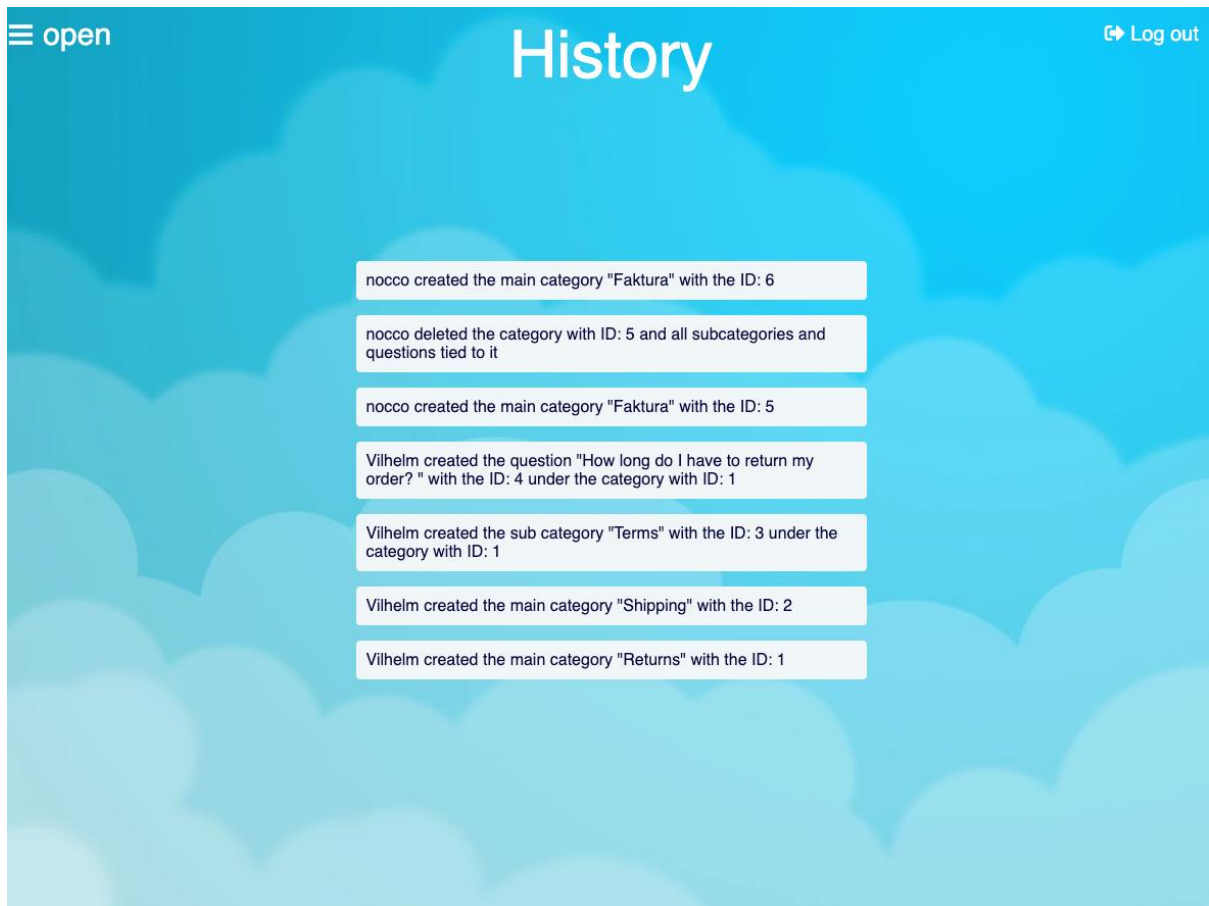


Figure 6.22: History page

There is also a page, seen in figure 6.23, where the user can change personal account information, such as email address, username, and password. Just like for the create account page, here it also exists a function validates the text in the email-field and makes sure it is written as a correct email format, if not a red banner will appear and the changes will not be saved to the database. If everything is filled in and written correctly as in figure 6.23, and the changes have been saved, instead a green banner will appear.

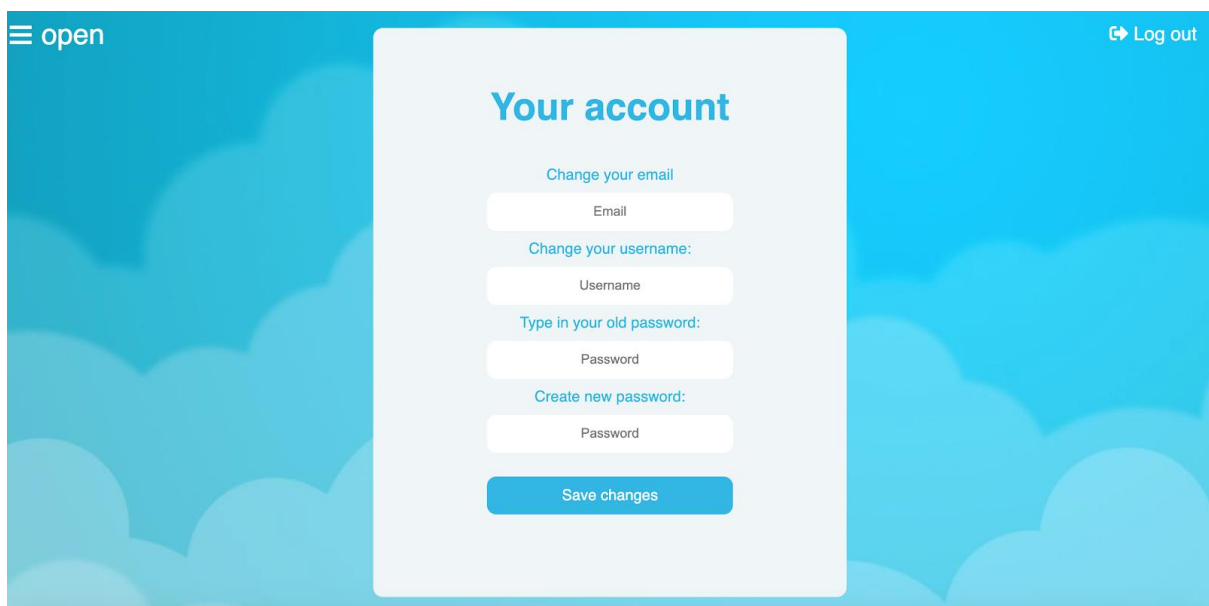


Figure 6.23: Changing the account settings

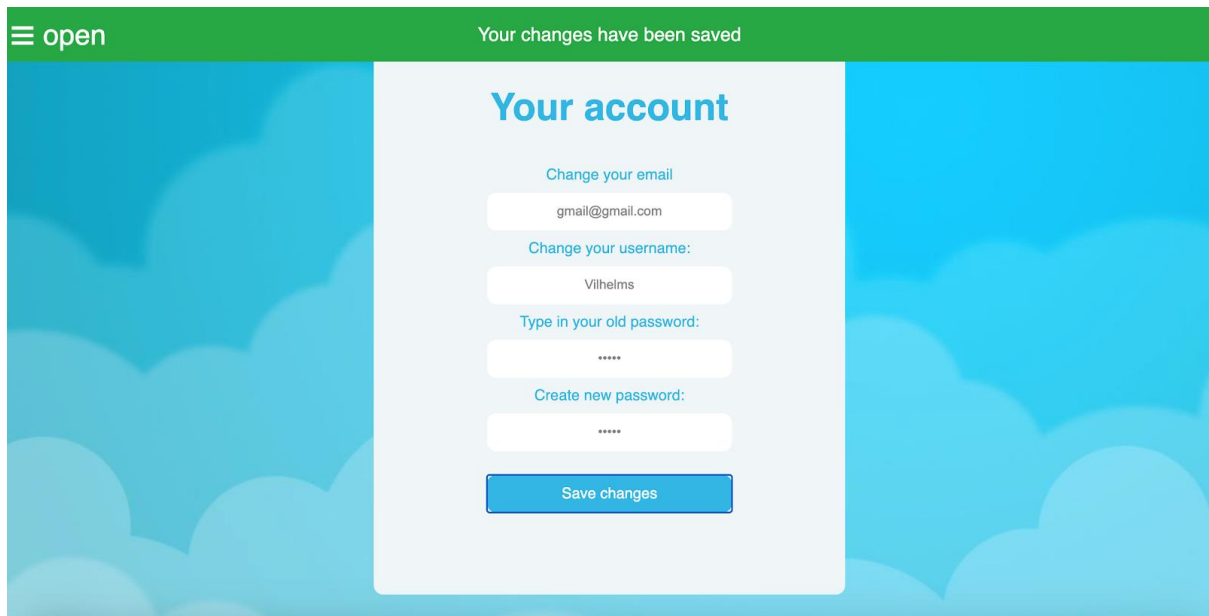


Figure 6.24: Changes has been saved

6.2 Frontend design

6.2.1 Chat

As frequently mentioned, lightweight and scalability were the essential themes throughout the chat-project, due to the requests from Connectel of the chat being easy to implement and resource-efficient. These requests did not only affect the backend coding and the structure of the chat but also the design, mainly because of two reasons.

Reason number one is based on the limitations or at least the difficulties for implementing certain design features when working with plain JavaScript and CSS without any libraries or add on: s. The possibilities to accomplish the same features libraries provide of course exist, but it is a lot harder and more time-consuming. For us working within a short time frame with a strict deadline, a limitation had to be made.

Reason number two is related to reason number one but is about a decision being made at an early stage, that the chat design of the chat should reflect and signify the lightweight and scalable structure it is built on and thereby keeping it stylistically pure.

6.2.2 Chat - Usability

When creating an interface, it is important to keep in mind the intended users of the application and design the interface to fit the users. For our customer service chat, the intended users, in most cases, are first time users who encounter the chat for the first time.

It is therefore important that the chat is easy to use and navigate and does not require any “training” to use. This is something that we have had in mind during the design process. This fits very well with the previously mentioned design ideas of keeping it simple and stylistically pure.

We have also tried to keep the chat as standardized as possible. This means that we have placed buttons and features in places where they usually appear and thereby feels normal for the user. For example, if you as a user were to close the chat, where would be the first place you would have looked for this function. For many, it would be either of the top corners because that is the standard place where this kind of function is placed, and the user do not need to spend time searching for the function. This way of thinking also includes the icons being used.

6.2.3 Chat - Color Scheme

An important factor for good design is the color scheme since it sets the tone for the application both for the professional appearance of the application but also how different colors have different impacts on our mind, associating colors with emotions.

The first thought for the choice of color scheme was to make it customizable via the management page, so Connectel’s customers individually could change the colors to match the chat with the customers own websites. This was unfortunately not implemented due to lack of time.

When then choosing our own color scheme we decided to go with a spectrum of blue colors with the main color matching a frequently used color by Connectel.

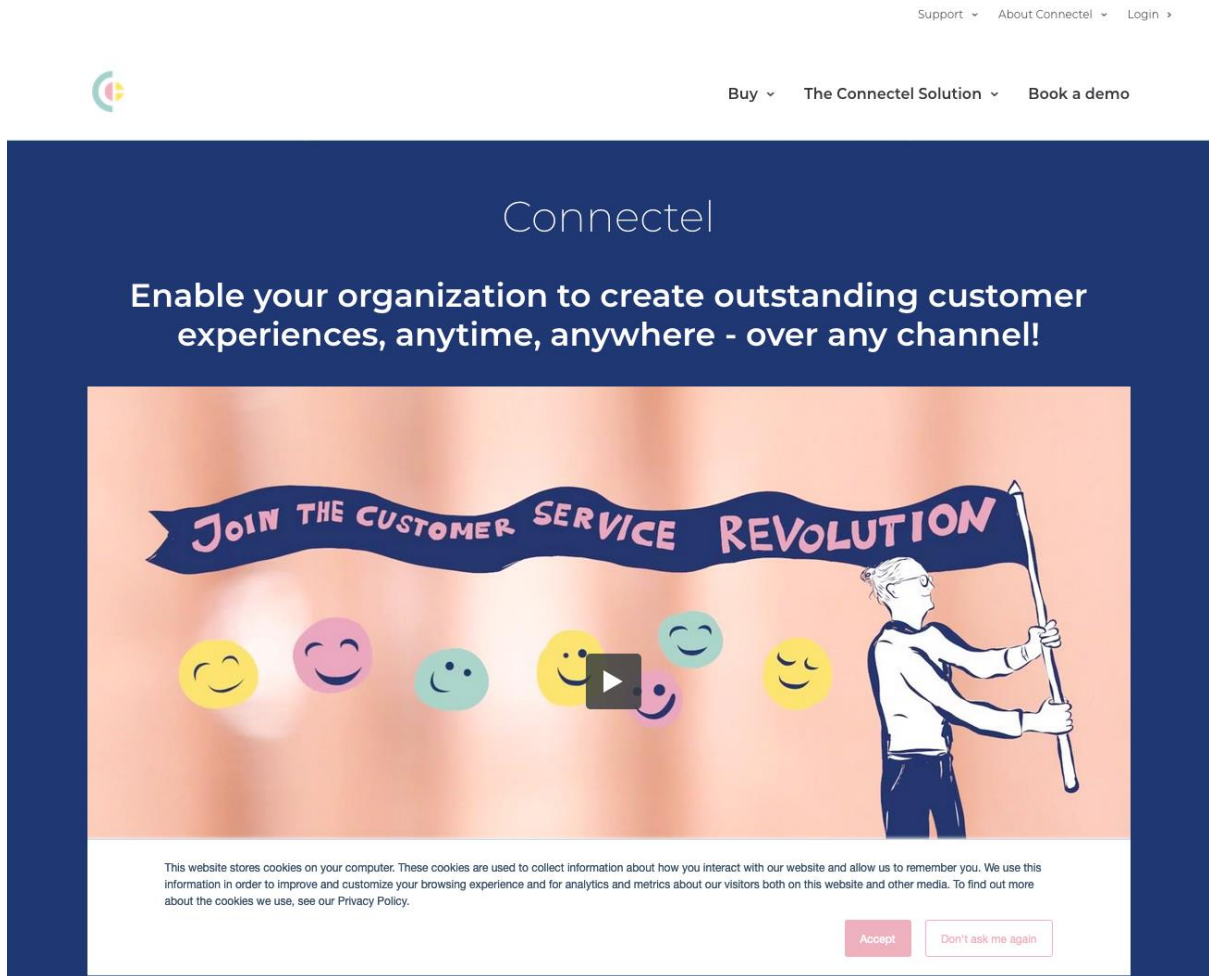


Figure 6.24: Screenshot of Connectel's Website (4)

But the decision was not solely made on the purpose of color matching with Connectel since a lot of customers will use the chat, which most likely using a different color scheme for their company than Connectel. The decision was also made since blue is emotionally associated with *Trust, Wisdom, and Knowledge*. Associations well suited for a customer service chat. (7)

6.3.4 Management

The interface for the management part was designed to create a modern, vivid, and floating workspace. This was accomplished by using light colors, a simple and clean font, and adding some figures in the background, in the shape of clouds.

6.3.5 Management – Usability

Since the intended user for the management part of the product is managers at customer services, the system does not have to be as simple as the chat. This is since they are not classified as first-time users since the managers have the opportunity to learn and get educated within the system. The interface can thereby be more complicated since the user

learn how the system operates. In our project, we have kept that in mind but still made the use of the system straight forward. With Easy to navigate between the different sites. Easy layup allowing for Connectel to add features of their own.

6.3.6 Management – Color Scheme

Likewise, as with the chat, the original idea was for the customer of Connectel to be able to choose their own color scheme, matching their company's. But just like with the chat this feature had to be skipped, due to lack of time. Instead, we at first choose to use the same blue color scheme as for the chat but we realized, well implemented, it felt way to dull and dark, so we changed. We still wanted to keep the blue but decided to lighten it up with a lighter blue, creating a more playful atmosphere.

7. Backend-development

When it comes to the communication in both the management-page, ChatBot and the messaging-function we used the structure known as REST API. REST API stands for Representational state transfer and it uses the built-in functions of HTTP which is the application protocol of the World Wide Web. In our application we use the functions GET, POST and PUT which are receive, send, and update objects, respectively. How we use REST API in the ChatBot, and messaging differs slightly and will be explained thoroughly individually.

The management-page and ChatBot is linked to the same database which locates in MySQL. In our project we all use our own localhost as server which means it is essential in the current state of the application to have an own MySQL server on computer and also that all changes done with your application is only saved locally.

7.1 ChatBot

In the ChatBot-application the REST API is used to communicate with the MySQL-database where structure behind all the paths in the ChatBot-decision tree are stored as well as the data for management history and the users. In the management-site the structure is created by the use POST-request to insert, update, and delete data in the database to create the desired structure of the decision-tree for the company. The ChatBot itself then uses the database to illustrate the options in the decision-tree in the chat window by the GET-request which you can see in figure 7.1. The previous mentioned management-site also uses two more tables in the MySQL-database to illustrate the management history of the decision-tree and to keep track of the information of the users.

```
app.get("/getAll", function (req, res) {
  var categoryQuery = "SELECT * FROM routes;"
  connection.query(categoryQuery, function(err, results1) {
    if (err) throw err;
    var answersQuery = "SELECT * FROM answers;"
    connection.query(answersQuery, function(err, results2) {
      if (err) throw err;
      res.json({routes:results1, answers:results2});
    });
  });
});
```

Figure 7.1: The code that retrieves the information about the decision tree in the database and sends it to an URL where the ChatBot-application can use it

7.1.2 Routes and answers

The ChatBot uses two tables in the database to store the data of the ChatBot-tree, routes, and answers, which you can see in figure 7.2. Routes is the table where all main categories, subcategories and questions are stored. Answers are just as it sounds were all the answers to the questions are stored. In the figure below you can see the relation between the two.

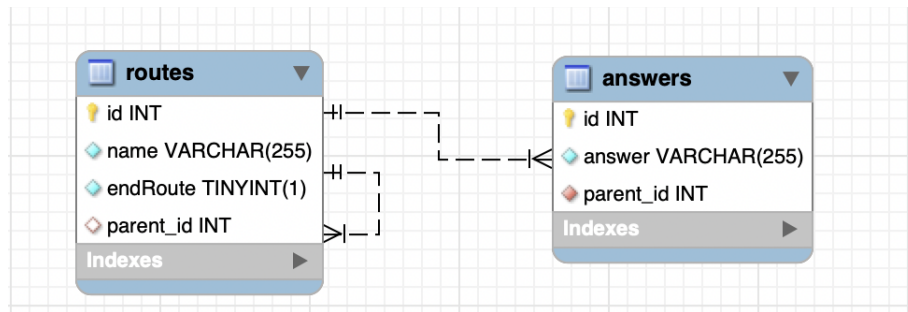


Figure 7.2: The SQL-tables routes and answers

Starting with routes, the rows in the table have information in 4 columns which are id, name, endRoute and parent_id. The column id is the primary key in the table which means it is unique, not null and the identifier for the row. In addition to that the id is auto incremented which means the column is filled automatically when created. This feature benefits the user of the management program in the sense that the user does not have to input a unique number when creating the row which in a company with a need for a largely populated table is very helpful. The column name is the input of the user of the management-page what the category or question should be named and must also mustn't be null. The column endRoute is value of true or false which indicates row is a category or a question. The column parent_id is a foreign key which references another rows id and the column is used for indicating which row in the table is the parent. The other table, answers, contains information about the answers to all the questions in routes. This table is populated with three columns which are id, answer and parent_id. Id and parent_id in answers works in the same way that id works in routes and answer is the column which the answer string is stored.

As previously mentioned, the ChatBot-tree are populated by 4 different objects which are main categories, subcategories, questions and answers and are located in the table's routes and answers. The tables are all populated in through use of the management-page which communicates with the MySQL-server by the HTTP-request POST and node.js express. When pressing on a submit-button, confirm-button or delete-button on the management-page, the input the user is be sent forward by a POST-request to the file server.js which then inserts, updates or deletes one or more rows in the MySQL-database. There is 4 various ways and row is created in the database depending on which object the user wants to create. If a main category is created in routes the parent_id and endRoute is set to null and false since the main categories are the starting points of the decision tree with no parent nodes and should have subcategories and questions as child nodes. The code for creating a main category can be seen in figure 7.3. When creating a subcategory or a question in routes the inserted value in management page is set as name and the parent_id is set to the id of row it belongs to which could be a main category or another subcategory. The difference between creating a subcategory and a question is the value of endRoute is set to false when creating a subcategory and to true when creating a question. In addition to that when creating a question an answer is created simultaneously in the table answers by taking the id of the question as parent_id and the input value of answer as answer. In the management-page you could also update and delete the created objects using said objects unique id. If you delete a category with child nodes attached to it all nodes beneath should also be deleted. This is achieved by a MySQL-command when creating a foreign key in a table called "DELETE ON CASCADE". What it does is to delete all rows which have to deleted nodes id as a foreign key

which then also leads to that all rows that have the deleted child nodes as parents also gets deleted.

```
app.post('/createMainCategory', function(req, res) {
  var category = req.body.categoryName;
  if (category) {
    var insertionQuery = "INSERT INTO routes (name, endRoute, parent_id) VALUES (?, false, NULL);";
    connection.query(insertionQuery, [category], function(err, results) {
      if (err) throw err;
      var action = "CREATE MAIN"
      var actionQuery = "INSERT INTO management_actions (userId, username, action, subjectId, subjectName, objectId) VALUES (?, ?, ?, ?, ?, ?)";
      connection.query(actionQuery, [req.session.userId, req.session.username, action, category], function(err, results) {
        if (err) throw err;
        res.end();
      });
    });
  }
  else{
    res.end();
  }
});
```

Figure 7.3: An example of an HTTP-POST request that inserts a main category in the MySQL-table routes

When then the tables have been populated the ChatBot is ready to be used by the client. When the user has entered his or her details in the chat window the ChatBot automatically starts. The API then uses various types of GET-commands and if-statements depending on the user's decisions to display the correct data in the decision-tree to the user. An illustration of this can be found in the figure below:

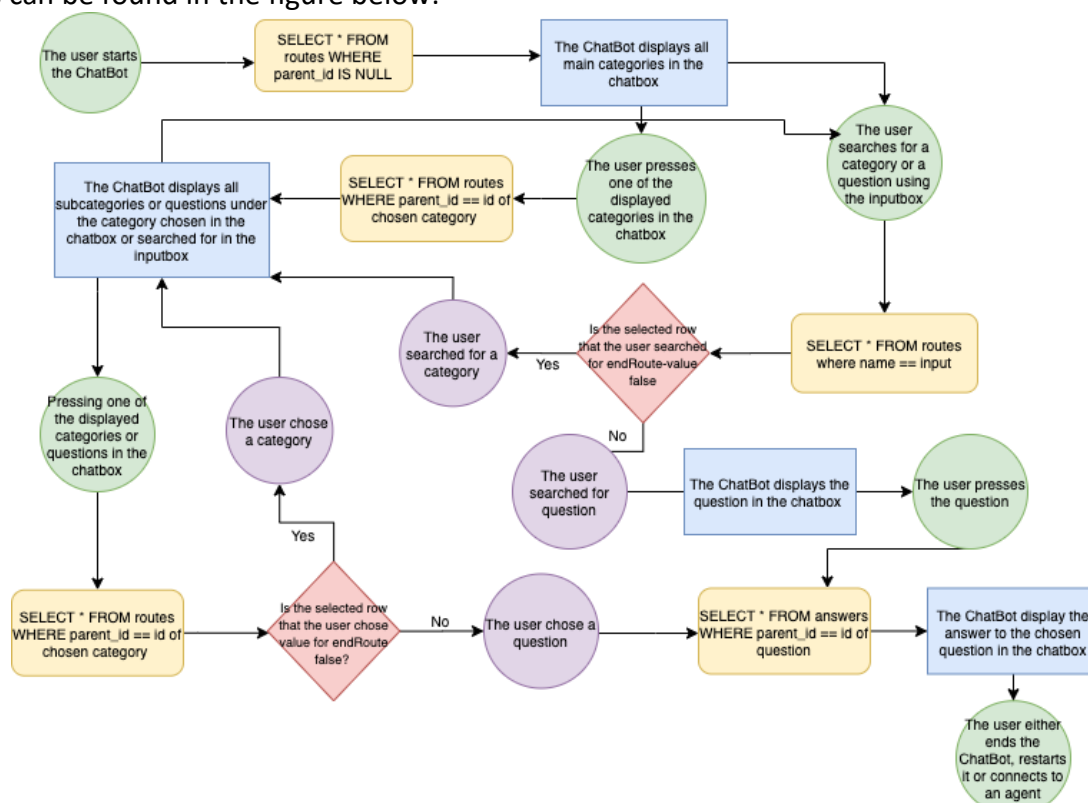


Figure 7.4: A flowchart of how the ChatBot retrieves its information from the MySQL-database

7.2 Management

In 8.1 we briefly glossed over the backend communication between the management-page and the MySQL-database in the case of populating the tables used by the ChatBot application.

In this chapter we are going to present the other two tables in the database, users and management_actions, which are being used by the management-page and the communication behind the interface and the database.

7.2.1 Users

To use the management-page you need to login to the website an using an account. All information about a specific account is stored in the MySQL-table users, which you can see illustrated in figure 7.4. The five different columns of information in the table us userId, username, email, password, and manager. The userId is the primary key and is automatically set when created just as the primary keys in routes and answers. Username, email and password are the user input inserted when creating an account. Since you cannot know your userId in the creation of the account the email-column is also set to unique so that there cannot be two account with the same email. The password is stored in the database as a hash value because of security reasons which will be described in bigger detail later. The column manager is a Boolean value which is the indication to whether the specific account is a manager or not. This column was inserted in case that we would create an agent-view on the website for co-workers in the company without manager privileges but is now always set to true.

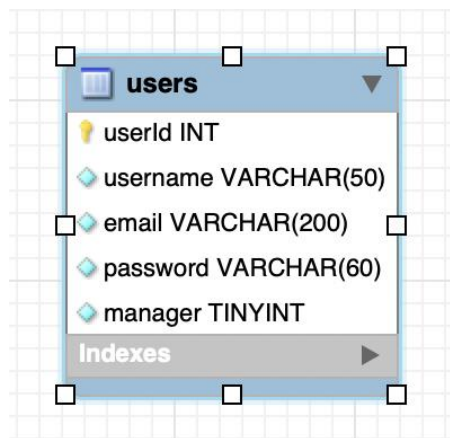


Figure 7.5: The table users in the MySQL-database

When an account has been created in the database you are now able to login in with the information inserted in the table. The user puts in his or her account details in the specified fields on the login-site. Through a POST-request it searches for the input-value of email in the MySQL-table. When a matching row is found the input value for password is encrypted in the same way as in the creation of the account and compared to the hash-value in the database for the matching row. If the hash-value is the same in the database is the same as the hash-value of the user input in login a session is created where the row's value for username and userId is set as the user information, the status of the session is set to logged in and the user is redirected to the start page of management. If either a matching email is not found, or the password does not match the user will be sent a message that the email or password was wrong. How the session-handling works will be more explained in the security section.

In the management-page there is also a way to update the profile which the user is logged into. The POST-request for this is seen in figure 7.6. This is done by taking the logged in users id and updating the columns in the specific row with the user inputs. To be successful when

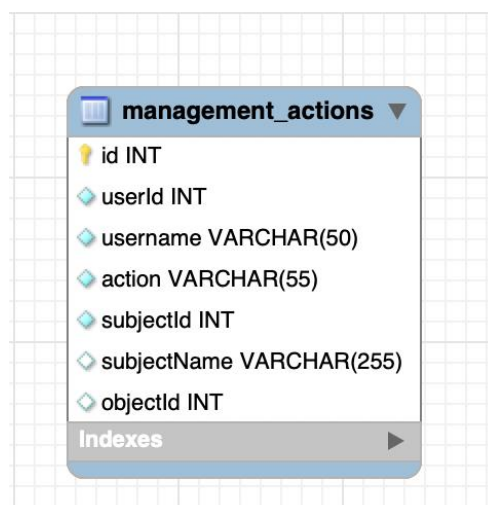
updating your profile, you must input your old password. If the inputted password does not match the password in the database, the update is not done in MySQL.

```
215
216 ✓ app.post("/updateAccount", function (req,res){
217   var userId = req.session.userId;
218   var newEmail = req.body.newEmail;
219   var newUsername = req.body.newUsername;
220   var oldPassword = req.body.oldPassword;
221   var newPassword = req.body.newPassword;
222   ✓ if (newEmail && newUsername && newPassword){
223     var loginQuery = 'SELECT * FROM users WHERE userId = ?;'
224     ✓ connection.query(loginQuery, [userId], function(err, results) {
225       if (err) throw err;
226       ✓ bcrypt.compare(oldPassword, results[0].password, function(error, cryptResults) {
227         if (error) throw error;
228         ✓ if (cryptResults == true){
229           ✓ bcrypt.hash(req.body.newPassword, saltRounds, (error2, hash) => {
230             if (error2) throw error2;
231             var updateQuery = "UPDATE users SET email = ?, username = ?, password = ? WHERE userId = ?;"
232             ✓ connection.query(updateQuery, [newEmail, newUsername, hash, userId], function(err2, results2) {
233               if (err2) throw err2;
234               req.session.username = newUsername;
235               res.end();
236             });
237           });
238         }
239       ✓ else{
240         res.send("Incorrect password, return to account-page")
241         res.end();
242       }
243     });
244   });
245 }
246 ✓ else{
247   res.end();
248 }
249 });
```

Figure 7.6: The POST-request that update the information in the table users of the logged in user

7.2.2 Management_actions

The table management_actions, which is illustrated in figure 7.7, is used for the history page on the management page. The content of it you can see below. This table is populated with rows that describes a certain action done in the management-page, which in backend terms means that the rows describes changes done to the routes and answers



management_actions	
id	INT
userId	INT
username	VARCHAR(50)
action	VARCHAR(55)
subjectId	INT
subjectName	VARCHAR(255)
objectId	INT
Indexes	

Figure 7.7: The table management_actions in the MySQL-database

This table is populated with rows that describes a certain action done in the management-page, which in backend terms means that the rows describes changes done to the routes and answers. Because of that, the table is populated right after a query is done concerning the routes and answers. The id is auto incremented and the userId and username, which is taken from the information in the server session, describes which user made the changes. Action is a string that is the indicator which type of action that was done in the management-page and the subjectId and subjectName is the id and name of the row in routes that is being inserted, updated or deleted. If a subcategory or a question is being made in the management page the following row inserted in management_actions will also have an objectId which is the id of the parent category of said question or subcategory. The POST-request for adding a new question and the following query for creating a row with the information about the change in management_actions is seen in the figure below.

```

329
330 app.post('/addNewQuestion', function(req, res) {
331     var question = req.body.question;
332     var answer = req.body.answer;
333     var parentId = parseInt(req.body.parentId);
334     if (question && answer && parentId) {
335         var questionInsertionQuery = "INSERT INTO routes (name, endRoute, parent_id) VALUES (?, true, ?);";
336         connection.query(questionInsertionQuery, [question, parentId], function(err, results) {
337             if (err) throw err;
338             var selectQuery = "SELECT * FROM routes WHERE id = LAST_INSERT_ID();";
339             connection.query(selectQuery, function(err, results2){
340                 if (err) throw err;
341                 var questionId = results2[0].id;
342                 var answerInsertionQuery = "INSERT INTO answers (answer, parent_id) VALUES (?, ?);";
343                 connection.query(answerInsertionQuery, [answer, questionId], function(err, results3){
344                     if (err) throw err;
345                     var action = "CREATE QUESTION";
346                     var actionQuery = "INSERT INTO management_actions (userId, username, action, subjectId, subjectName, objectId) VALUES (?, ?, ?, ?, ?, ?);";
347                     connection.query(actionQuery, [req.session.userId, req.session.username, action, questionId, question, parentId], function(err, results4){
348                         if (err) throw err;
349                         res.end();
350                     });
351                 });
352             });
353         });
354     }
355     else{
356         res.end();
357     }
358 });

```

Figure 7.8: An example of a POST-request that populates the routes and answers tables with a new question and answer and also creating a row in management page with information about the insertion

7.3 Messaging through HTTP requests

The following section will outline how the messaging within the chat has been handled. The chat solution which we partly inherited from Connectel and applied on our application uses polling and HTTP requests. These terms will be further explained below, but first we need to establish what “HTTP request” means. HTTP is short for Hypertext Transfer Protocol and indicates communicative actions such as transferring of HTML documents to be performed via internet. (15)

7.3.1 POST

A POST request refers to the sending of a request to insert an entity or change the state of an existing entity which results in effects on the associated server. (15)

7.3.2 GET

A GET request refers to the fetching of an entity or state that is located on the associated server. GET is used only to receive resources and not change them. (15)

7.3.3 PUT

A PUT request changes the state of an entity that is located on the associated server. PUT is used only to change an existing entity. (15)

7.3.4 Polling and messaging

Polling means continuously updating and checking the state of something.

Polling is done on both the agent- and the client side. This is done because the pipeline containing the messages is shared between the user and the agent. Polling was done through the JavaScript built in function `setInterval`, which takes a parameter for how often you want to perform something, in this case, the parameter is 1000 which translates to 1000 milliseconds, or 1 second. In this function we make sure to call all the required functions for continuously check for messages and tokens (the technicalities of this will be explained further below). The basic principle of sending and receiving messages is the following:

1. A message is posted through a POST request; the message is given an ID (this ID is related to earlier messages)
2. The application polls the server with a GET-request with a time interval (in our case, one second, i.e. every second it tries to GET the latest message from the API).
3. The application receives the earlier posted message with the GET and displays it in our chat application.

Every time a message is sent, it is done so with an ID that relates to earlier messages that has been received. Figure 7.10 shows the payload of the message containing both the interaction id and a unique message id. This message id is fetched and incremented each time a new message is received, see Figure 7.11. Every time a new message has arrived, a new URL is also given to the URL to receive only the latest message in the queue. This can also be used when reloading a page to keep track of what the latest message is and to fetch all the previous messages that the interaction contains. In this specific case, the URL containing `id="+id"` the "id" is set to 0, to be sure to receive all the messages that have been sent in the interaction.

```
function startChat(cookieId){
  document.getElementById("proxy").remove();
  document.getElementById("send").setAttribute("onclick", "postConnectel();");
  document.getElementById("message").value = "";
  if (!cookieId){
    postToken();
    createLoader();
  }
  else if (cookieId){
    getMessagingHistory(cookieId);
  }
  myVar = setInterval(function() {
    getToken();
    getConnectel();
  }, 1000);
}
```

Figure 7.9: Polling through setInterval

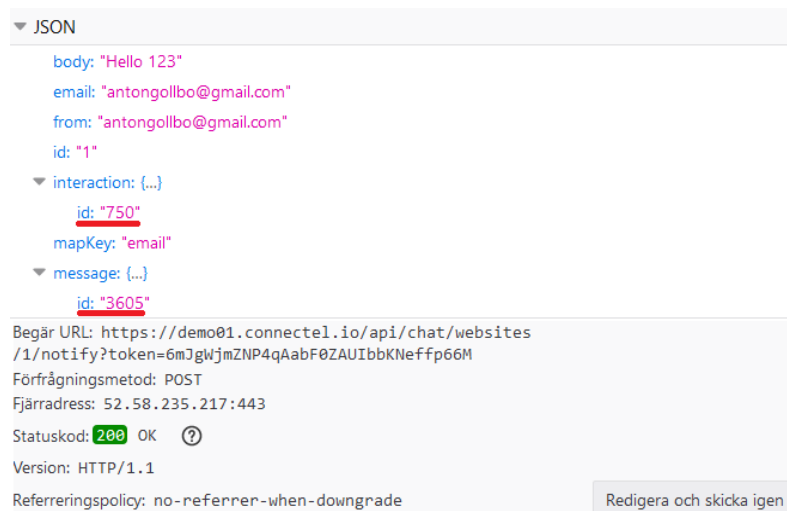


Figure 7.10: HTTP POST message

```
function postConnectel(){
  var cookies =document.cookie.split(';')
  .map(cookie => cookie.split('='))
  .reduce((accumulator, [key, value]) => ({ ...accumulator, [key.trim()]: decodeURIComponent(value) }), {});
  axios.POST("https://demo01.connectel.io/api/chat/websites/1/notify?token=6mJgWjmZNP4qAabF0ZAUibbKNeffp66M",
    {"body":document.getElementById("message").value,"id":"1","mapKey":"email",
    "from":cookies.email,"email":cookies.email,"message":{"id":cookies.inputId},"
    interaction":{"id":cookies.interactionId}})
  .then(function(res){
    document.getElementById("message").value = "";
  })
  .catch(function(err){
    console.log(err);
  });
}
```

Figure 7.11: HTTP POST message in code where the cookies are remapped and saved to a variable and then used together with the inputted message to send the message to Connectels agent-site



Figure 7.12: GET message from the URL with interaction id and message id

```
function getConnectel(){
  var cookies =document.cookie.split(';')
  .map(cookie => cookie.split('='))
  .reduce((accumulator, [key, value]) => ({ ...accumulator, [key.trim()]: deco
deURIComponent(value) })), {});
  axios.GET("https://demo01.connectel.io/api/chat/interactions/"+cookies.inter
actionId+"/my_messages?id="+cookies.inputId+"&includeAgent=true&token=6mJgWjm
ZNP4qAabF0ZAUibbKNeffp66M")
  .then(function(res){
    var convo = res.data.rows;
    if(convo.length > 0){
      document.cookie = "inputId="+convo[convo.length-
1].id; //SETTING THE INPUT-ID COOKIE
    }
    lastMessageIndex = convo.length; //IF A NEW MESSAGE IS SENT OR RECEIVED
THIS VARIABLE IS OVER ZERO
    if(lastMessageIndex > 0){
      var newObjectList = convo.slice(0,lastMessageIndex);
      drawChat(newObjectList); //HERE THE MESSAGE/MESSAGES IS DISPLAYED IN C
HAT
    }
    else if (lastMessageIndex <= 0){
    }
  })
  .catch(function (err){
    console.log(err);
    agentCloseChat(); //IF A AGENT CLOSES THIS FUNCTION WILL THROW AN ERROR
- AND THIS FUNCTION IS CALLED
  });
}
```

Figure 7.13: GET message in code that checks that a new message has arrived. If it has, the code sends the new messages to drawChat() that displays the new messages in the chat box.

7.3.5 Sending of files

The chat application supports sending of files. This is done through a similar way as the messages are sent although some differences are present. What happens first is the document that will be sent has to be uploaded, this is done through the API and by sending the document in a JavaScript object type called “FormData” in a POST request. This is shown in figure 7.13. This first POST also returns a response variable which is called “attachment id” to know which file belongs to which link. This attachment id is then used in the next POST request, which sends the link to the file as a href element which can then later be inserted as a HTML element wherever you want on the page. The second POST request is shown in picture 7.14 and 7.15. This file is then displayed in the chat window as a special message which indicates you can press on it to download the file. This is shown in figure 7.16.

```
Begär URL: https://demo01.connectel.io/api/chat/interactions
/838/attachment_upload?token=6mJgWjmZNP4qAabF0ZAUlbbKNeffp66M
Förfrågningsmetod: POST
Fjärradress: 52.58.235.217:443
Statuskod: 201 Created ⓘ
Version: HTTP/1.1
Referreringspolicy: no-referrer-when-downgrade
```

Redigera och skicka igen

Figure 7.13: Attachment upload through POST request

```
Begär URL: https://demo01.connectel.io/api/chat/websites
/1/notify?token=6mJgWjmZNP4qAabF0ZAUlbbKNeffp66M
Förfrågningsmetod: POST
Fjärradress: 52.58.235.217:443
Statuskod: 200 OK ⓘ
Version: HTTP/1.1
Referreringspolicy: no-referrer-when-downgrade
```

Redigera och skicka igen

Figure 7.14: POST request for link to file

```
▼ JSON
AttachmentId: "530"
body: "<a href=\"https://demo01.connectel.io:443/api/chat/interactions/838/attachment_download?attachId=530&token=6mJgWjmZNP4qAabF0ZAUlbbKNeffp66M\" target=\"_blank\">1.PNG</a>"
email: "antongollbo@gmail.com"
from: "antongollbo@gmail.com"
id: "1"
▼ interaction: {...}
  id: "838"
  mapKey: "email"
▼ message: {...}
  id: "4303"
```

Figure 7.15: Attachment contents

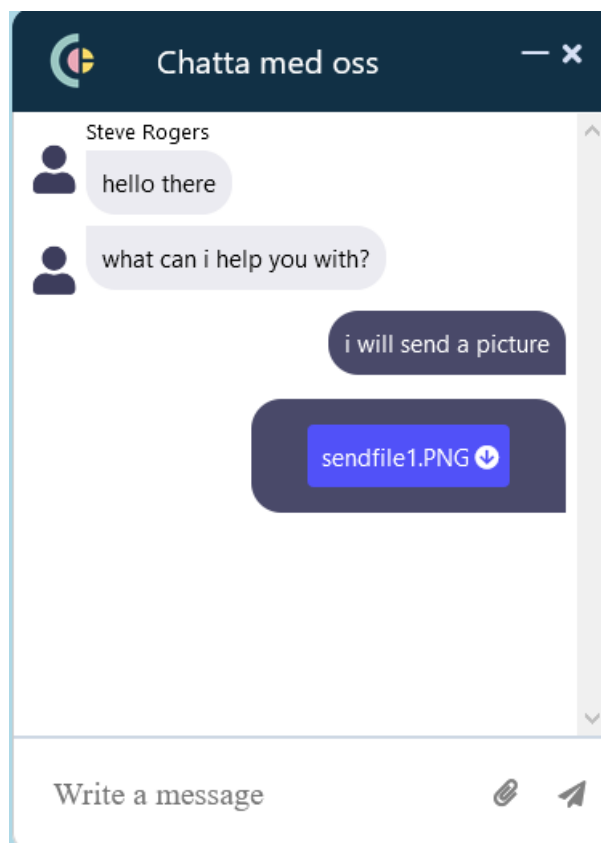


Figure 7.16: A file is sent in the chat

7.3.6 Token verification through GET and POST requests

The communication between agent and client uses a nominal token system. This token is first received when connected to the agent and is used to identify the current conversation between agent and client. In the application this is referred to as a “interaction id”. In figure 7.17 the initial payload and URL the POST is performing is shown containing information about the client connection such as IP and email. How this is done in code is shown in figure 7.18, where the parameters email, interaction id and name are saved locally for the browser

session in cookies (more on this later). This message is sent to the URL of the API and the response is shown to the right, giving the established connection or “conversation” an interaction id. The token system is also a way to secure the communication, making sure that these interactions ids are never used again and that they expire after some time. The tokens are verified through GET requests every second, this is shown in figure 7.19. A GET message is here sent to an interaction id specific URL which contains the conversation with the given interaction id. This is done to verify that the chat is using the right token and accessing the correct conversation and messages.



Figure 7.17: Token posting and receiving interaction id

```
function postToken() {
  var cookies = document.cookie.split(';')
  .map(cookie => cookie.split('='))
  .reduce((accumulator, [key, value]) => ({ ...accumulator, [key.trim()]: decodeURIComponent(value) }), {});
  axios.POST("https://demo01.connectel.io/api/chat/websites/1/notify?token=6mJgWjmZNP4qAabF0ZAUlbbKNeffp66M",
    {"body": "Name: " + cookies.named + "\n" + "Email: " + cookies.email + "\n", "referer": "https://connectellabs.com/", "customerIp": "130.243.237.212", "id": "1", "mapKey": "email", "from": cookies.email, "email": cookies.email})
  .then(function(res){
```

Figure 7.18: HTTP POST request to receive unique token. The body contains name, email and customer ip.

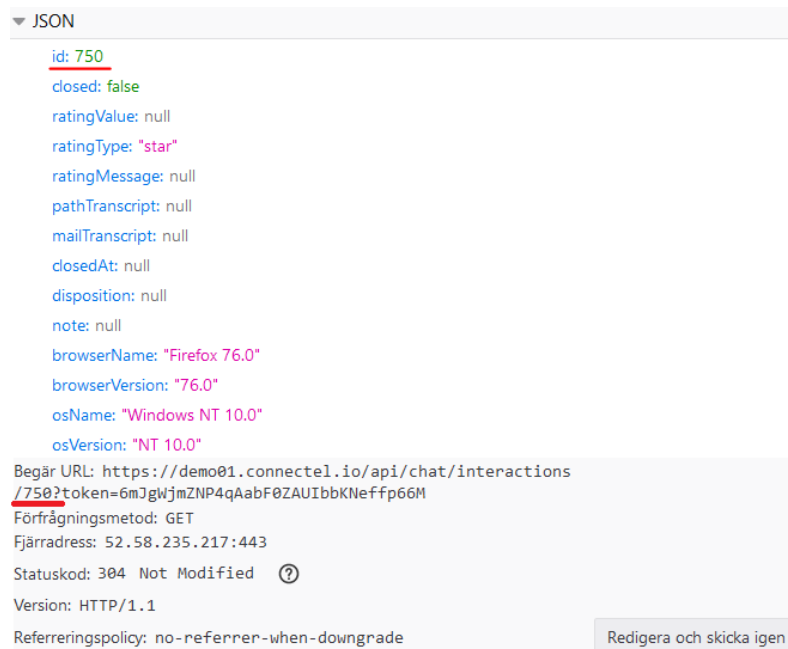


Figure 7.19: Token GET

7.3.7 Cookies

The definition of a cookie is the following: “An HTTP cookie (web cookie, browser cookie) is a small piece of data that a server sends to the user's web browser. The browser may store it and send it back with later requests to the same server. Typically, it is used to tell if two requests came from the same browser — keeping a user logged-in, for example. It remembers stateful information for the stateless HTTP protocol.” (16)

Cookies was used in this project to be able to save state of the chat interaction. The application currently uses several cookies. This is done as previously stated to save state but also to avoid using global variables in our code which is generally known as bad practice. How the cookies are used will now be demonstrated and explained.

In figure 7.20, the first possible state is displayed. In this state, the browser has not saved any cookies to contain details of the user and chat session. This is the initial state of the chat when it has been opened in a new browser session for the first time.

In figure 7.21, the second possible state of the chat is displayed. In this state, the user has entered its name and email which both have been saved in the browser session in a cookie. If the user now refreshes the site or opens a new tab and visits the same address, the state will have been saved and the name and email is still stored.

In figure 7.22, the third possible state of the chat is displayed. In this state, the user has entered its name, email, and generated interaction id and input id. These variables have been saved in cookies and if the user refreshes the page, they will be stored and used the next time the chat window is opened.

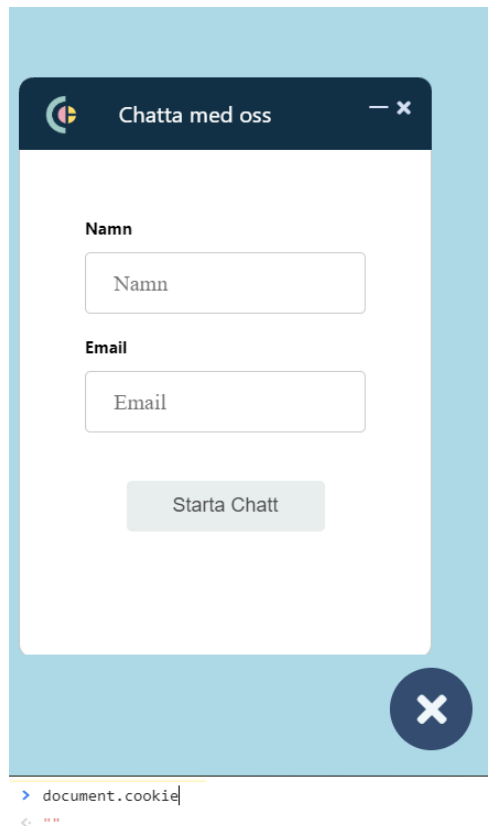


Figure 7.20 – First cookie state

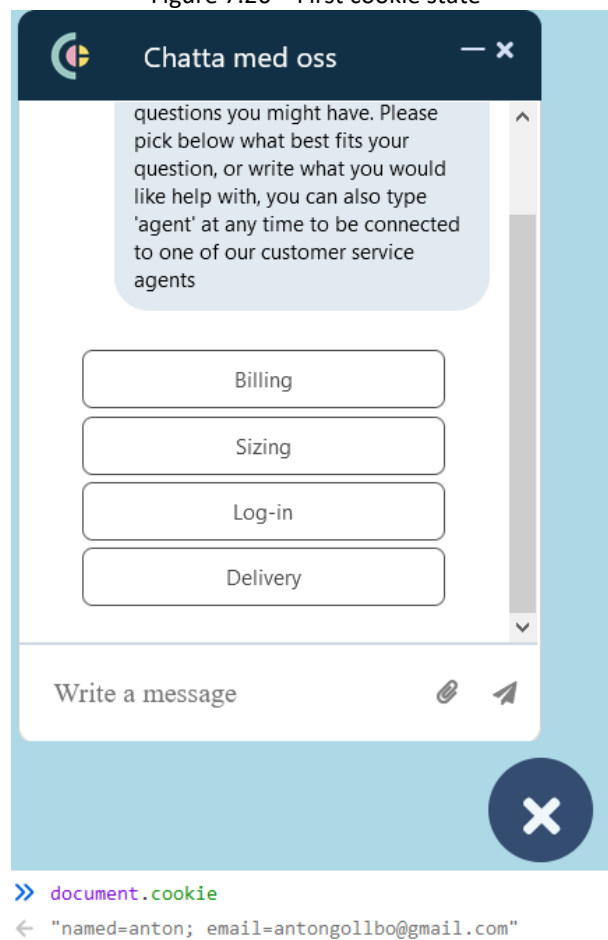


Figure 7.21: Second cookie state

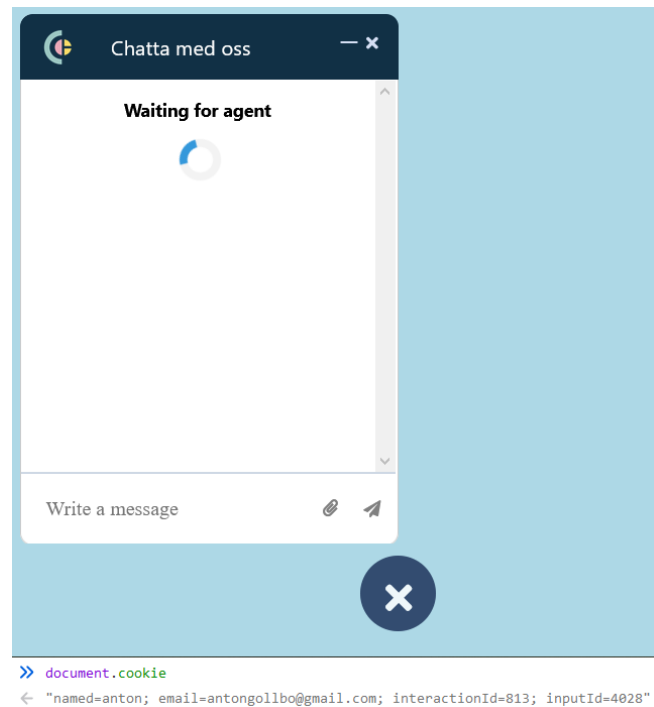


Figure 7.22: Third cookie state

7.4 Security

7.4.1 Authentication

The management page is as said made for the managers at the company which means you must authenticate yourself that you have access to the website. To reach this goal, we have used previously mentioned MySQL for the database part with all the information about registered users and a Node.js package called express sessions to handle the authentication process. When logging into the management-page the server compares the user input with the users in the database and see if there is a user in database that matches the values inputted at email and password. When they do match, a so-called session is created using express-session and you are then redirected to the starting page in the management page. The login request can be seen in figure 7.23. A session is an abstract term for a created object in the server that keeps track of information about the user. For example, it is in the session `userId` and `username` is saved for the logged in user which then is used in the creation of rows in `management_actions`. The most important function of the session is to give access to the management page. If you try to go to any page part of the management site without having a session started with access you will be redirected to the login page. This keeps non logged in users to have access to the site. This can be seen in code in figure 7.24.

```

178
179 app.post('/loginPost', function(req, res) {
180     var email = req.body.email;
181     var password = req.body.password;
182     if (email && password) {
183         var loginQuery = 'SELECT * FROM users WHERE email = ?;';
184         connection.query(loginQuery, [email], function(err, results) {
185             if (err) throw err;
186             console.log(results);
187             if (results.length > 0) {
188                 bcrypt.compare(password, results[0].password, function(error, cryptResults) {
189                     if (error) throw error;
190                     if (cryptResults == true){
191                         req.session.loggedIn = true;
192                         req.session.userId = results[0].userId;
193                         req.session.username = results[0].username;
194                         res.redirect('/start');
195                         res.end();
196                     }
197                 });
198             }
199         });
200     }
201 }

```

Figure 7.23 – The login-request that sets up the session-details if the user and password is valid

```

73
74 app.get("/management", function (req, res) {
75     if (req.session.loggedIn) {
76         res.sendFile(path.join(__dirname, 'views/management.html'));
77     }
78     else{
79         res.sendFile(path.join(__dirname, 'views/login.html'));
80     }
81 });
82
83 app.get("/managementHistory", function (req, res) {
84     if (req.session.loggedIn) {
85         res.sendFile(path.join(__dirname, 'views/managementhistory.html'));
86     }
87     else{
88         res.sendFile(path.join(__dirname, 'views/login.html'));
89     }
90 });
91
92 app.get("/account", function (req, res) {
93     if (req.session.loggedIn) {
94         res.sendFile(path.join(__dirname, 'views/account.html'));
95     }
96     else{
97         res.sendFile(path.join(__dirname, 'views/login.html'));
98     }
99 });
100
101 app.get("/start", function (req, res) {
102     if (req.session.loggedIn) {
103         res.sendFile(path.join(__dirname, 'views/startmanager.html'));
104     }
105     else{
106         res.sendFile(path.join(__dirname, 'views/login.html'));
107     }
108 });
109

```

Figure 7.24 – The handler for all the views in management that only redirects you to the site if the session status is logged in

Another part of the login process that helps the security is the hashing of the password before it is stored in the database. This is done with a module in node.js called bcryptjs. When a user creates an account, the server inserts the inputted information the MySQL database. In the case of the username and email the information is inserted just as inputted but the password is first made into a hash value using 6 salt rounds and then inserted along with username and email which you can see in figure 7.25. When then the user wants to log in the inputted password there is also made it a hash value and then after that compared to the hash value in the database to see if it matches. This precaution is done because the password should not be able to be seen and figured out by looking into the database. For example, one manager should not be able to see another managers password.

	userId	username	email	password	manager
►	1	Big Boss	ceo@shoes.se	\$2a\$06\$GYn3SRhr/OphGGETW/XufinkwPJsh...	1
	2	Marcus	marcus@shoes.se	\$2a\$06\$vX2a6MxictJN9AoP3nvAJ.CLKCYJ7D...	1
	3	gmail	gmail@gmail.com	\$2a\$06\$8SGSZ1Ccn/Kxl/3CO5lavugY4uwyVdP...	1
	4	gmail	gmai@gmail.com	\$2a\$06\$0/R1j/nAXlu/VANU3KUvKOEVBhNgCd...	1
	5	tjokatt	tjokatt@gmail.com	\$2a\$06\$6xUCIIJwKx3Q2E1Pt6vxCOhaNwguU...	1

Figure 7.25 – The populated table users with the hash value stored in the column password instead of the actual password

7.4.2 Security risks

There were some parts in the security aspect we did not have time to address and could be improved upon. One of these is the fact that in the current state of the application, anyone that runs the application can create an account that gives access to the management page. If we were to move the application from working locally to a company for example, a precaution that prevents non-managers from creating an account. This could be by possibly using a code in the creation process that is uniquely created and which gives permission in the server to create a manager account in the database.

Another possible risk lies in the structure of making an insertion in the table management_actions. A row in management_actions should be created simultaneously as a query is performed to route. The code however first does the query to routes and then does the query to create insert the action into management_actions. If the server would throw an error on the second query the action to routes itself would be done but the row that should be inserted in management_actions that describes action would be lost. This would then mean that the user would not be able to see the action report in management history. Another example of this with a more serious outcome is the creation of a question where the question information is created in routes first and the answer information is created in answers is created second. If the second query throws an error here the question will be without an answer and the ChatBot would be affected.

8. User testing

When it comes to user testing, we decided as a group to only test the Client chat including the ChatBot and exclude to test the management page. This decision was made because of the user profiling that says that the management page should be used after training and to GET useful results we would have had to train our test subjects before testing them and we decided that it wouldn't be worth the time.

When it came to the testing of the Client chat, we experienced that most of the features were understood by the test subjects. Manoeuvring the ChatBot using the buttons and chatting to an agent were no problem at all for the people that tested the app. This could depend on the fact that a chat is something we a large majority of people us in their everyday life and have a lot of experience with.

The major comments on improvement we received was in the use of the AutoString function and how to connect to an agent. When searching for a subcategory the child nodes for the subcategory is shown which some people thought was a little confusing when not seeing the actual word hey searched for in the chat window. This was something we thought about but decided to not implement but after the testing we realized that could have been a better idea. How to connect to an agent was the second comment we often got. In the app the user should write agent and send which is described in the first chat bubble. Since the chat bubble is rather long some people said they missed it and would like to have had a go to agent button instead. This is something we did not consider because it would go against our design plan in keeping the chat as clean as possible.

9. Personal reflections

9.1 Marcus Löthman

This project has been a learning experience which I right now cherish the absolute most through my 16-year long journey in the Swedish school system. These weeks I have for the first time in my academic career spent all my time on a single task and because of it I have learnt so much about myself, working in a team, programming and much, much more. In this personal reflection I am going to go through certain aspects of the project and try to explain what I learned from that specific aspect and in the end I will summarize all parts to a personal conclusion of it all and how I will use this experience in the future.

To start it off, I am going to talk about the technical part of the project, in other words the programming aspect and learning experiences in the making of the product. This project is the first time in my time at Uppsala University where I have gotten the opportunity to make something out of nothing. In other courses tied to programming the task have either been really small or in case of a big task, you would always receive a skeleton-code to understand and build from. In this project you started from a blank canvas with no pointers on where to start. That was the first real learning point. The fun was to explore different ways of making a website or an app and reading up on how different they are and the pros and cons of them all. This gave me taste of how many different paths there are to go and an insight in the world of web- and app development. The vast amount of choices also led to a challenge, which was in which way to go to begin to project. Because of the blank canvas start there are infinite projects to start with and in addition to that there are also many frameworks to use in your chosen project. We knew before starting that we wanted to create something useful that possibly could even be used at a company or by the public. Therefore, our starting point were to reach out to different companies to find a path to start in which we found at Connectel. They helped us with tips on how to start the project and what we should use to reach our intended goals. This approach I think worked very well in our favour. To be humble and look for different sources with experience in fields relevant to your project for information on how they would handle it I think minimizes the risk for setbacks when working and it will definitely be something I take out from this project.

After we had gathered a plan in how the programming part should be done, we started working directly. We all worked simultaneously with the coding but in the beginning, I focused a lot of my work with the backend and specifically database handling in MySQL through Node Express. This part of the project I found very rewarding because I got to expand on my previous knowledge in databases. Before the project we had completed one course in database handling where we used MySQL. In that course we went over how databases are built and how you should be structured to work in the best way, but we never really got to test it out in a big project. This project gave us an to do just that on our own. In the development we tried many different structures and as time went and our project demanded more and more out of the database and we had to constantly update and improve our work to adjust to the new demands. If I remember correctly, we changed the structure of the database behind the ChatBot in a big way at least 2 times. Firstly, all categories, questions and answers were located in the same table which gave us trouble when working with creating

the ChatBot since we needed the application to distinguish between the different objects in a clearer way. After that we tried having questions and answers in a separate table but that didn't satisfy our demands either so we changed it again to the final solutions with objects that are displayed as buttons in the ChatBot in one table, routes, and the objects that are displayed as answers in another table, answers.

We also got to explore the many different options and functions in MySQL when working with the database creation. Examples of this are plenty, for example in the beginning we had trouble and how the id should be inserted in the database, so the program recognizes it in a simple way. The problem originated in the problem that different users should be able to insert a row in say the routes-table without knowing which id there are in the table. This was solved by the function auto-increment on the column id when creating and solved a lot of our problem. There were a lot of these small findings such as "last_insert_id()" and "delete on cascade" that helped our application to run smoothly. Both the big structure rebuilds and smaller changes in the database creation have taught me a lot. In the process we did not really sit down and really write down all requirements for our final database but rather we started creating the databases right away and rebuilt it for the demands along the way. Of course we could have been helped by planning more properly from the start but I believe I learned a lot from reevaluating, rebuilding and exploring the features of MySQL and I now feel much more confident in my knowledge of database design now that I have actually been part of developing one from the scratch.

Further along in the project more of my focus shifted to the web design using plain JavaScript and CSS along with working on making sure the HTTP-requests functioned correctly in the chat. Starting with the web design of it I am now after the project very pleased that we decided to use plain JavaScript in our project. The decision was made in order to fulfil one of our goals to make the application lightweight, but I believe it worked out in our favour. At the start, I found it kind of hard to be working with JavaScript because I felt that it in order to create something really small required you to write many lines of code. Furthermore, after that you had only created the element, if you wanted to look like you wanted you had to do an equally amount of work in CSS. I haven't got any experience in other frameworks in web design but from what I understood, using plain JavaScript is a quite demanding way to create a website and I really believe my experience with it will help me tremendously. Now I feel after all hours spent in JavaScript and CSS like I have gotten a real firm grasp on how to use it and because I have gotten this training without any other frameworks will be a real asset in the future when moving over to using some framework. To summarize this programming language part, I feel like our experience have laid me the foundations of web design which in the future will help me when learning new frameworks and ways to develop applications and websites.

In the last leg of the project development my time was for the most part focused on the communication between our chat application and the Connectel's agent site which was made through HTTP-requests. Earlier in the project I had worked some with these requests in the communication between the ChatBot/management-site and the MySQL-server. This task though was more challenging because there already were requirements on how you should connect, send and receive messages to the agent site which meant that we first had to look up how that worked. This part of the project was personally for me the part where I had the most fun and found most rewarding. To look into the polling system with GET-request from

the agent site to receive and displayed the conversation and how to send a message and connect to the site through a POST-requests and saving some of the responses in cookies was a whole new chapter for me which I hadn't touched upon before. I felt like it was here where I gathered the majority of the brand-new knowledge in this project and working with it gave me a desire to learn even more in the future. I felt like looking into how Connectel old Client chat worked network wise and then imitating that and succeeding gave me a whole new understanding of the Internet and its possibly the greatest lesson learnt technically in this project for me.

Now I am going to leave the technical part behind and focus on the reflection on what it was like working as a group in this project. Firstly, I should point out that all four of my co-workers in this project are all 4 of my absolutely closest friends and we were before the project started already a close group. This meant that some things came for free, like chemistry and knowing each other's personalities, and some additional challenges. Because we knew each other it was sometime hard to distinguish between work hours and leisure hours. Some lunches were a tad bit long, workdays were cut short at times and some days we probably could have been more effective. While that may have been the case in the beginning, I think we developed through the project and began to find a way to progress in the project at a good pace. I also believe that what we may have lost at the start in effectiveness we covered in excellent communication where we had very little miscommunication that held us back.

Continuing on the theme of the work progress, this was my first real experience with agile methodology and although I was sceptical at first, I now understand why it is being widely used. My initial scepticism was grounded in the fact that you had to spend so much time outside the work itself, like updating my Trello board, and thought that it would steal needed time from the programming part. In retrospect I now feel like it has helped me tremendously. To spend 10-15 minutes and set up goals what you should is a great way to inspire you to move forward and illustrate the problems of the project and when you complete a task it inspire you to keep going even more and that helped me to stay focused all spring. I also was helped by the fact that even though we had our own tasks within the project we would all the time put them aside for a while to help other members of the group. This method I found very helpful because it helped me get a picture of all aspects of the programming, got me motivated when I sometimes had given up on a certain task and would boost the chemistry of the team when completing a task as a duo or team.

On other aspect essential in our project was the company side of it all. As said before we worked with the company Connectel as sort of consultants and this was a new position for me. This was first and foremost a very positive experience where we had a very well working relationship with communication about their requirements for the product and our questions. At times we felt stuck and always when we felt that way, we could communicate our situation to Connectel, and we would be able after that often continuing our process on the right track. On the other hand, working with a second party did sometimes pose us additional questions. For example, the biggest setback for us was after we had pitched our initial project to the university, Connectel told us that the project we were planning on performing wasn't possible to do at that moment which made us start over from square one. A similar situation arose

when we after a week of working with web sockets got the information that the chat must communicate with polling. These two events really tested our group mental strength and even though we felt very defeated in those moments we overcame those obstacles and prevailed which I think that was a real nice lesson about working with second and third parties. It may not always go exactly how you think but, in those moments, it is important to adapt to that and move forward in the new direction as soon as possible.

In this reflection I have so far focused mostly on the positive sides and before I conclude, I am going to take up some things the group and I could have done differently. When it comes the group, we started too late with the work. Even though we knew that we wanted to work with a company we did not make any preparation before the period with the project started so we had to start scramble in finding a company in the first two weeks. To prepare properly is of course something I already knew but, in this project, I found out once again how stressed you can get when you are not ready from the start. On a personal level I found out that I probably give up to quickly on tasks I feel are tough. There were a couple of instances where I thought that a specific task was impossible, and I put that task on the backlog because I did not want to deal with it. Then most of the time one of my co-workers started working on it and pulled me into the process and which always led to success. I now know that I can achieve tasks in programming that seems way above my paygrade if I give it a proper try which is one of the most important things I have learnt this spring.

Now to summarize, I must return to my original statement in the beginning of the reflection that this was the most rewarding project I have ever done. To work on my programming skills where I got to evolve skills I already had and learn whole new ones while at the same time getting the experience to work as a team full time for a company is the closest you can get to simulating a work experience in the academic world. All the things I mentioned above are experiences which I will take with me in future projects and work and I hope to do similar projects in the future.

9.2 Vilhelm Söderström

What have I learned:

I believe that this course have been the most fun and instructive course during my three years studies at the University I really appreciate the fact that the outlines of this project have been very free and independent and I think that is one of the reasons why I really feel like I have learnt a lot during this period. The freedom has made us to act and seek information ourselves which I believe has forced us to learn things otherwise would have been missed out. I also enjoyed the project has really since it has felt like something you would meet outside the university in the real working life since it has contained what I have understood as common and important tasks. Working in a team, starting at zero and then commonly create a plan and a together work towards a common goal. To get that sort of experience at the university I believe is valuable since it in some way differs from ordinary university life. Other courses of course include parts of this but not the same extent. You often get some kind of foundation to build the project on and is just not as comprehensive. So to get this kind of experience, I believe prepares you for the working life coming. I especially have appreciated the training in collaboration, something this project really required which has been consistent throughout

the period. From deciding what to do, creating a plan to achieve this and then working towards a common goal. This is a quality you always can keep evolving within and to get to train it within the extent this project has demanded really has given me a lot and made me develop.

The decision to work with a company is something I am very happy about since I believe it was a valuable learning opportunity. I can imagine it is very similar to work as a consult for a firm and therefore an instructive experience for those kinds of professions. I believe I got a better understanding on how a “customer - provider” relationship works out in real life and how to deal with all the struggles and emotion that follows. We had some frustrating incidents where we, due to insufficient information and miss in communication had to throw away solutions we had been working on for a long time. One example of that was the solution for the chatting. We thought that we could use a socket solution but later found out that the preexisting system used a polling solution, which meant or time spent on the socket solution was a complete waste. This really made me realize the importance on communication. Expect, the pros I have already mentioned about learning and gaining experience, working with a company has many pros whatsoever. For example, at times when we been stuck on a solution for a longer time, we could contact the company to get some help to keep going forward.

From a more technical point of view I have learned a lot of web programing and elaborately the skills previous courses within this subject has given me. I think it has been good to have the gotten opportunity to focusing on just one thing during a so long time period. Unlike previous courses it has given the opportunity to really develop deeper skills where you really feel like you have deep knowledge within the subject. This is something I have really missed from my previous university time where it has felt more like I have received a shallow knowledge within in a lot of subjects. This is something that actually made me a bit nervous for the working life after university due to lack of expertise. After this course I feel a lot more confident since the project has required to create a project from the foundation which has forced me to get a more profound knowledge unlike previous courses and project where you received some skeleton code to work with. This is a setup I do not prefer since I feel like you miss out on a lot of understanding which makes the work more difficult. I prefer to really start with the basics and work your way up. Not being thrown in working with different libraries and frameworks when you do not know the basics properly. So, in this matter I have really enjoyed our project and setup where we have worked with the basics, HTML, vanilla-JavaScript and plain CSS. This has made a lot of things more complicated and time consuming than if using frameworks. But I believe it has been worth it since I and actually feel like really have learned CSS, JavaScript HTML from the basics and now have great skills within them and then also feel more prepared to work with frameworks in coming projects.

My role:

My role in this project has in some meaning varied throughout. When we started this project the initial task for everyone in the group was to commonly come up with a project we all wanted to do. We all directly agreed on that we preferably wanted to do the project as a collaboration with a company, so the initial step was to search for a company to work with. To find a company as fast as possible we all took on the role to contact companies. When we then found a company to work with and the project description was formulated, we tried to

figure out the different parts of the project and divide them amongst ourselves. The project was divided into mainly two part, frontend and backend and my role in the project was set to frontend developer. The frontend, was planned to handle only the chat part of the project, including chat functions and design. While the backend was planned to handle the database structure and therefore also including the ChatBot functions. Throughout the project it was however discovered that this distribution would not fully work since a lot of parts of the project needed work in both the backend as well as the frontend to be implemented. The tasks, my main role as a frontend developer came to deal with was mostly about building the interface and the visible functions tied to it. Like making certain features appear/disappear and action to occur, when triggered. But also, to design the different parts of the frontend parts of the project. I therefore worked mostly with the appearance of the chat and functions tied to it but also a lot with the log in and management pages.

Challenges throughout the project:

Working with a project of this type in a group consisted of close friends like we did had a lot of upsides, but also some downsides. Sometimes I believe that it maybe became too much of a “hangout with friends” feeling which had its negative effects on the work itself. Instead of working, it was a lot of chatting with each other. This also spilled over to the break times, resulting in some maybe a bit too long breaks. Although this may have been the case, I also feel the need to emphasize the perks of us working in a group of close friends. I truly believe that it helped us in the communication within the group where all of us felt confident enough to talk to each other and express our own opinions. This I am sure resulted in less loss of good ideas and thoughts that may been lost in a less close-knit group.

The biggest set-back we had during this project I believe was the fact that we at the start of the period, started to work on a different project idea, than the one we ended up. This idea was formulated together with the company but had to be scrapped due to problem from the company’s side. This resulted in a pretty big time loss, seen to this limited time frame of the project. This set back, I believe symbolizes very well what I believe has been the biggest challenge with this project; the difficulties of working with an outside party. I already covered why so I am not going to repeat myself.

When it comes to the technical challenges throughout the project, we had our fair share. The issues that affected when we switched from our original socket solution to the polling solution for the chat. This resulted in a new way of thinking which took some frustration, time and research to get acquainted with. To be forced to learn new ways of coding was pervasive throughout the project but what made this part harder was lack of information to find of how to handle. Otherwise it was quite simple to found information on how to deal with a problem, but this was more problematic and took us a lot of time and effort. Otherwise in the beginning I struggled a lot with the CSS positioning part for different elements. But during the course of the project I got a better understanding of how it works, and by the end this was not a problem anymore.

Conclusion:

As a conclusion I am very happy and proud of what my group accomplished and produced during this project. We started off with a blank sheet and managed to create a chat solution for a real IT-company in plain CSS, JavaScript and HTML, using no frameworks or libraries. A chat solution, which the company also were content with. I really feel like I have learned a lot of web programming from the basics mostly thanks to the way our project was formulated and I would not change that if I got a chance. I also believe that I developed in and learned a lot of working together with people. Both in the meaning of working together in group but also working together with a third party, like Connectel. It is very demanding and really force you to learn to compromise and understand that everything is not going to work out like it may been planned from the beginning. It is, though, also rewarding due to several reasons.

9.3 Vera Widmalm

General thoughts

This period of my studies has been one of my most intense learning experiences so far. I really enjoyed working with my group, learned so much about programming, and got much more personal responsibility to manage working towards deadlines. As I mentioned, it has been a very giving experience, but the project has also come with some challenges that was hard to tackle. I am therefore very thankful to my group, the tutors on the course, and last but not least our contact persons at Connectel that has been very generous in helping out when problems arisen.

File structure

In previous courses, I would like to say that I have felt kind of “thrown in” to a new programming language quite abruptly every time. The last programming course I took was about JavaScript programming with a user perspective and was very intense and fast moving. Therefore, I am very glad that I with this course have gotten the possibility develop these skills and got a deeper understanding of developing web applications in JavaScript. For example, concerning the file structure of the project. I now know how each file references each other, and how to reach variables and functions from different files. Particularly in the frontend-related files, since I worked mostly with them. Briefly, the file called Indexscript is the content of the chat window, like a HTML-file with divs and buttons and so on, but transcribed to a plain JavaScript file. The file called tree.js is the file that loads the IndexScript- file if its chat button is pressed. The only thing this file contains is therefore the button, and sometimes a loaded iframe, to make it as lightweight as possible, but more on that later.

I also have gotten a better understanding of why and how it can be tricky to reach variables or features in other files, for example in the parent or child- file. An example of this was when I tried to implement the close- button in the top right corner of the chat window, which is located *inside* the iframe. This was hard just because the button inside the iframe was supposed to close the whole iframe, which would be easier to do from outside the iframe. In this case, the chat window acts like a “child” of the outside content, “the parent”. This was solved by making the button in the indexScript file trigger a “postMessage”, that was sent to the parent. Hence the parent listened to this POST message and removed the whole iframe if it received such a message.

Team management

When working with this fulltime type of project, I quite early realized how important the group dynamic is. Although this was not a *problem*, I really learned how important it is to cooperate and working in your team roles. The greatest lesson though, is how important it is to have the possibility to go outside your assigned role and work more floating with the project everywhere, helping all teammates. As I earlier mentioned, I worked mostly in the frontend, but thought it was very giving to get to help someone in the backend. It was very valuable to get help from different teammates with different perspectives and roles, and I think our team succeeded with this part very well.

Working with Connectel

Since we worked with a company, Connectel you also got a perspective on how you work with web developing at companies “in real life”, since I only have been programming things for school up to this point. It was interesting to see how much the full stack developer knew about programming and to tackle different problems. He always had very good input on that to do and how to solve different issues. I also think we all got some good experience on how a project is developed concerning goals and delimitations etc. We had not that much deadlines from the company, and pretty much structured our own timeline of the work, which of course also lead to some good experience on how to structure the project.

Data & Scalability

As we have explained, one key goal of the project was coding the chat without any frameworks, libraries, or bootstrap etc. This was done to minimize the loaded data when opening the chat. The loading was also not allowed to be done until the user wanted to chat, which I was part of developing and fixing. When working with these restrictions I learned how important it is with minimizing the size of the data in order to make the chat scalable and used by many at the same time.

As mentioned before, there was some trouble with reaching different variables in different files but got solved after a while. Working with loading iframes was also new, so this took a while to do, but gave me a lot of experience. Generally, I think I have gotten a view on how libraries etc. could be convenient to use, but also come with loading big data. I also hope I got a better understanding on how the code really works by coding in plain JavaScript. This since it is what the libraries are built on.

Design

When developing the project goals, CTO Jens Leijon was clear on saying that the user friendliness is an important part of the work. To achieve this, the design and the application structure plays a key role. For example, the different icons in the window has to resemble functions that feel intuitive for the user. The window has to have a “clean look” with no confusing add-on’s that throws him or her off. Jens explained that we could ourselves determine which functions we thought was more relevant and useful to implement.

Concerning colors and the design, we also got some references to other websites that have succeeded to build simple and user-friendly chats. I learned that a nice way of making the chat simple for the user, is to have a simple design structure with “lowkey” colors in the same theme. We kept all the icons simple and the color scheme very basic and clean. One goal within the design was also making the chat fitted for mobile users. Since the screen is so much smaller, the chat should be shown in the whole window instead. This is also why we had to implement the minimize- button in the top panel. The mobile users do not see the round minimizing button outside the iframe when they view the chat in full screen mode. Together with Siavash Goudarzi we decided that tablet users though should have the same view as computer users, since the screen is big enough.

Code hygiene

With this project I really learned how important the program structure and variable names are. When navigating through several hundred lines of code it’s very frustrating if you don’t find the right variables because you made a sloppy job naming them. Hence, we also realized that structuring the code after functions, order of appearance etc., really eases the work when cooperating or working with someone else’s code.

The big picture

Even though this course has taught me a lot about working in a project team generally, the biggest part of my learning experience has been about programming. I have really developed my skills in JavaScript and CSS, but also my technical skills in problem solving and how to look up information about different functions on the internet. I would say that I now have dealt with a lot of beginner problems and know how to tackle new projects in the future.

9.4 Anton Gollbo

My experience with this project can be likened with the experience of playing a game of football. The overall experience is performing something I am passionate about and love to do, while certain aspects of it will be tough and challenging. Together with four of my best friends, we have tried to create something of value from the ground up. This has been an extremely rewarding and “eye-opening” trial and course of events.

The journey that we embarked on began the first week of the assignment through us contacting several companies in the IT-sphere in Uppsala. We wanted to try this approach since we had heard you could collaborate with a company and felt it could be an exciting way to take on this project. We received interest from several companies (does anyone want some free workers?) and worked out a few ideas with a few of them. The first idea was not very well received by our supervisors, so we decided to move on and try a different company. We got in contact with Jens and the company he represents, Connectel. He was very interested and hesitant on formulating a project that we could take on. The first proposed project was to create a live statistic mobile application for team leaders in customer service. The idea of

this idea was to create a visualization of how the customer service is “performing” live and to be able to take actions thereafter. We thought this was a great idea and something we wanted to do. We worked out a few of the details with Jens and then presented the idea to the supervisors who thought it would be a so called “smash-hit”. Unfortunately for us and everyone else, it was not to be at this point in time. The API architecture was not set up in a way for this to work smoothly with Connectel to create this application. We thought: “Alas, for thy shalt not hang thy head in vain, we shalt find a new way onward”, and so we did. We got in contact with Jens again and he proposed another project that they had “lying around”, which was to create their own chat web application. The company already had a chat through a system environment called “Xcally Motion”, although this was not their own chat and therefore it had some flaws that they could not affect. The idea was then instead to create their own chat application which could communicate with the API and “Xcally Motion” but done in a more effective and scalable way. We as a group chose to accept this challenge and work with this project.

I together with Marcus, formed what we called the “Backend”-team. Our initial prime goals were the following:

1. Create and establish a backend structure that consists of content for our “ChatBot” function and communicate through our server to the database.

2. Create the functionality within the “ChatBot” and “AutoString” functions in the project.

Backend-programming is not something I was familiar with prior to this project, although I was intrigued by the notion of it as it seemed technically challenging and a good way to lay a foundation to learn how technicalities work “behind the scenes”. I still quite do not know how to explain what backend is or where the line is drawn between back- and frontend. Although I have learned that it is of much importance to handle backend in a sophisticated manner, as large parts of functionalities will depend on the structure that you choose to have in your backend solution. For this, I am very happy that we chose to learn how to set up a “REST API”. At first, it seemed like a buzzword meaning basically nothing which we chose to ignore and started to build our server and database structure it was handled on our previous web application course. We realized after some struggles that this was not the way to go. In an application that relies heavily on data handling (which are a lot of applications) structure was of utter importance and insertion, extraction, deletion and updating were actions that we needed to perform smoothly. This is where we started to work toward a REST solution. As I said, this seemed like just another buzzword at first but later we found that the structure of a REST API filled the requirements we needed and wanted (I guess this is why it is one of the industry standards today). As soon as we made the REST solution, we understood how we would move forward with using our backend. The REST solution laid foundation for every part in our backend and created a smooth ground for our whole backend to stand on.

The biggest challenge we faced in our backend was the handling of messaging. At first, we wanted to create a solution which occurred locally and not relying firstly on the API we had available at Connectel. We did this through MySQL and insertion and deletion through our backend. This solution worked fine but had a few flaws, one of them being that it was not

very heavy for us to run. The challenge ahead would be tough, seeing as the API supplied was not very easy for us to understand with such basic understanding of HTTP requests. We asked for help from Connectel and their developer and he went through how the chat functions and how the HTTP requests are working within the chat. Although this was very helpful, we had a long way to go to make it work with our chat solution. First, we had to understand how the unique sessions are handled, which was through tokens and unique conversations ids. When we understood this, we discovered how we could access the messages contained in JSON objects through URLs in a browser. As soon as we understood this, something clicked, and a realisation was in the works that maybe this problem would not be unfeasible after all. The heavy work was now to understand how we wanted our client side to handle the communication and accessing the right messages and keeping them present on page updates and opening new tabs. Cookies solved this problem for us. Once again, we thought cookies would be something very advanced and highly technical, but we quickly realised after some research that it could be implemented in an easy and solid way. This could also help us with a big problem we had earlier, which was that we had implemented global variables (something we have been told from programming day 1 is a bad practice and should not be used if you can avoid it) which were used in every bit of HTTP request and did not quite know how to not use these. Cookies allowed for smooth storage and keeping track of interaction id's and message id's but also keeping state when the page refreshes. All of this felt like a huge victory for us, since every problem seemed insurmountable but using the information that one can gather and keep working at it we managed to resolve our issues and end up with a very acceptable solution. The dreary matter of this is that the work behind our solution does not show as anything cataclysmic because it is "just" handling of data. However, when we managed to solve this, we felt we had a solid application and project which we felt we did not have to feel ashamed of. When we presented our solution for Connectel they were surprised with what he had managed to achieve (at least that is what they said) which felt fulfilling and it is not very often in our academic careers where we had put so much effort into a project for such a long time and end up with an end product of not only what we had done, but also invaluable experience and knowledge. Having the role of being one of two backend programmers was interesting and made me leave with a sense of wanting more. I was very happy with this role and think I learned a lot, although I do not think it mattered very much seeing as the project is a big collaboration, everyone touched on every subject matter in some shape or form.

The project overall has been an interesting and worthwhile experience. I feel that the course being (if this is even to be called a course) so free helped to make this a memorable experience. The weekly meetings were good for the supervisors, but did not "help" us very much, which I assume is the idea. We worked within a group of five. This could be seen as too big of a group to collaborate within, but I think it worked perfectly fine. Especially since we divided the group further into the backend and frontend groups. Moving forward, I would like to give programming and web programming a bigger part in my life. I find that it is one of the purest forms of problem solving I have ever had the pleasure of doing. The craft or art of

programming is also something that intrigues me, and I am very happy that I got the opportunity to find something I am really interested and motivated in doing.

9.5 Sofia Lövgren

These past ten weeks have by far been the most educational of my three years at university. This is the first time we have had the opportunity to build something completely from scratch, which was kind of frightening but also very exciting. I have learned lots and gained invaluable experience in working in a group, collaborating with a company, and problem solving. “Learning by doing” describes exactly how these weeks have gone by. We all wanted to aim high, and to do so the collaboration and communication had to work. To develop is something I can see myself working with in the future, so I really saw this project as a good practice for that.

Collaboration with Connectel

First, I want to acknowledge the fact that we got this project by collaboration with the company Connectel. At the beginning of the project, when we were in the phase of deciding what we wanted to do, we all felt that it would be cool to do something for a company, that maybe actually can be implemented and used by them. We got in contact with Connectel, which offered us this project, as something they wanted to start developing after the summer, and they wanted us to set the groundwork for the application. We found this very inspiring and fun to do something that was sought after. It was interesting to see how an IT-company works and to get help from an experienced full-stack developer. It got us some pressure to really deliver, so we set our goals high. Connectel has been a mainstay, but we have been very independent during work. I found that doing a project for a company, got our motivation and goals increasing. I think we all got a better understanding how working life function and realized how important communication is. At first, we handled the chat messaging with WebSocket’s. We had come quite a long way with the work, but after having a meeting with Connectel, we realized that this was not the way they wanted the messaging to be handled. We found out that their pre-existing system used polling, so we had to change everything. I think we learned a lot from this backlash. We got more confident in asking questions frequently so we could get feedback and directions.

Team management

In my opinion, collaboration is the most important factor in making a project like this to work. I also think it was our groups most well-functioning feature. I got into this project with four of my best friends which I respect both privately and academically. Of course, we all were a bit worried in the beginning that it would be hard to stay focused when you are working with people you are so close to, but this was never a problem. We decided to have work hours, which worked great. To work with your best friends made the project a lot more fun, since we always have a great time together. I think it also took away a lot of prestige you might feel sometimes. In our team, you were not afraid to ask someone for even the smallest problem. We did split the group in front-end and back-end responsibilities, but these grouping were quite weak. Everyone jumped back and forth between responsibilities, since our

communication in the group were so good, so we all were always aware of what everyone was working on. When we realized we had to change our messaging handling to a polling solution, I remember that I started of trying to solve the problem by myself. It was quite hard to understand because it is not a standard way to handle messaging. After some research and realizing the method depends on HTTP-requests, I remembered that this was a method Anton and Marcus had used to send data to their SQL-database. I immediately found the function in their code and could use that as a skeleton for my solution. I knew this because everyone we all simultaneously showed and explained how we solved problems. This is a great example of how our communication did our work more flexible, and also an example of how we switched responsibilities now and then, since I mainly worked on the front-end side, but still could take on a task that was more on the back-end side.

This was also my first time working with Agile development, and I think it worked great. These kinds of methodologies sometimes seem kind of unnecessary to me because I think team structure and management often comes naturally and I do not want to spend unnecessary time on this. But in such a big project like this it, in a large group, and everything should be built up from scratch, it is important to line up all the goals a minor task that must be done. This made the work much more efficient, as everyone could always go back to our Trello Board, and see what tasks that must be done. Every member in the group always had something to do. In the beginning, I was chosen to be the project manager. We did not want so many roles, and we did not want them to be so strict either. We decided that the project managers main task was to keep a clear view of the work to be able to keep track of what should be prioritized and how one is in relation to the schedule. I believe we all did this in some matter, so my role as a project manager was not that strict or a workload. We all helped. This worked for us since we know each other so well, had good communication and cooperation. I think that in most other groups, it would have been more necessary to divide the group into roles and responsibilities.

Technical aspect

I want to talk a bit of technical part of the project. We all entered the project with a couple of programming courses in our backpack. The most relevant course for front-end programming was the interface programming course we read this autumn. The same period, we read Databases I, the only programming we have done in back-end coding. In all these courses, you were given skeleton code, which was very helpful, but also resulted in not understanding how everything works thoroughly. For example, I had worked with Node.js before, but I had no idea how it really functioned. I did not know how to set up a server file, something that is not that hard at all but because we were given all the basics from start, you never got into it.

We started this project with no restrictions or pointers at all. As I mentioned before, it was kind of overwhelming at first. After some research you realized how many ways there is to build up an application or website. We really needed to read carefully and see the advantages and disadvantages of the different approaches. When we then got in touch with Connectel, we got some guidance and some referral which tools and what approach we should have to reach our goals. One of the main goals, was to make the chat scalable and lightweight. The pages where the chat would be implemented on, often have a lot of traffic, and you want to avoid unnecessarily large loadings. Therefore, we could only use basic vanilla JavaScript to code with, along with CSS-code for the styling. We were a bit familiar with the language, but

in earlier courses and projects, we had used frameworks in JavaScript, Vue.js for example. These framework makes it a lot easier to code, the files do not get that long and the graphical design becomes easier to format, since you can import a lot. Using vanilla JavaScript, means that we have written exactly every code line ourselves. Nothing is imported or added as a framework. Since we were used to using frameworks earlier, we always compared the coding with our earlier experiences, so I think we all thought it was time consuming to only write in plain JavaScript. I remember when I had to translate all HTML-code to JavaScript. It is not hard to understand or to write, but it took so long time. 80 lines of HTML-code translated into approximately 300 lines of JavaScript. Now in retrospect, I think we all agree that it was the most educational we have done. To code in a lightweight language made our understanding much deeper and thorough. I can with great confidence say that I understand every line of coding we have done in this project.

Conclusion

This has been the most fun and educational course during my university studies. I have gained more experience working in a group, gained insight into working life and above all, developed my programming and problem-solving skills. Primarily I have learned how to program properly in vanilla JavaScript and CSS. One of the most fun things I think, is to feel that you have learned something during these years of studying, since in this project, you had to pick up previous knowledge and experience and actually use them in practice.

I hope I get the chance to work with this team again. It has been a great experience and I am proud of what we have accomplished.

10. Final Words

10.1 Future development

When having a finalizing meeting with Leijon and Goudarzi at Connectel's office, we presented our final product, along with a discussion about what is left for future development. The main areas included in this are listed below.

10.1.2 Less static design

We have developed a chat with a cleaner and simpler look, but in the previous version of the chat, the company representative had the possibility to change the color scheme of the chat. As of right now, the colors are hardcoded in the script, and the chat button not movable. We could not manage to do this within the time deadline we had but are positive to eventually implement this in the future if we had the possibility.

10.1.2 Customizable chat features

One thing that the group, and CTO Leijon agreed on in the beginning was that it would be nice for the chat to be customizable concerning which buttons and features that are displayed in the window. This was also not done due to time pressure but is also an area that we still think would be interesting to develop.

10.1.3 Customer support's opening hours

Right now, there is no difference of the chat depending on if there are available calling agents or not. Hence, there should be some kind of input for the company representatives to choose when they have customer support, and some type of message if there suddenly are no available agents, or if the chat interaction is placed in waiting queue.

10.2 Words from CTO Jens Leijon

When meeting with Leijon and Goudarzi we got good response of how the project as went. We asked for a written statement from Connectel and got the following text from Jens Leijon:

Sammanfattning

Arbetet har överträffat mina förväntningar. Jag är imponerad av hur bra det är gått och hur snabbt gruppen förstod och tog till sig uppgiften. Det har medfört att kommunikationen har fungerat mycket bra i projektet. Slutresultatet visar också att gruppen började att utforska möjligheterna med chatbotfunktionalitet som egentligen var option på nästa steg. Sammanfattningsvis är jag väldigt nöjd med gruppens engagemang och slutresultat.

Jens Leijon 2020-06-04

Figure 10.1: Project opinion from CTO Jens Leijon

10.3 Final thoughts

To summarize, we managed to reach our main goals by developing a chat application in plain JavaScript, that loads whenever the user presses a small button in the corner of the browser window. The chat also has a bot that welcomes the user in the beginning of the chatting and gives him or her alternatives on what their matter is about. Judging by all group member's personal reflections we can conclude that we all have learned a lot and thought that this was a very giving and interesting project experience.

11. Reference List

1. Wayback Machine, ShortSummaryOfRest. 2006. Accessed 2020-05-22, Available from: <https://web.archive.org/web/20060717120412/http://rest.blueoxen.net/cgi-bin/wiki.pl?ShortSummaryOfRest>
2. Statistics Sweden (SCB). E-handel allt viktigare för svenska företag. 2017. Accessed 2020-05-21. Available from: <https://www.scb.se/hitta-statistik/artiklar/2017/E-handel-allt-viktigare-for-svenska-foretag/>
3. Hedström, Linda. Guide för lyckad e-handel. Svensk Handel. 2011. Accessed 2020-05-21. Available from: <https://www.svenskhandel.se/globalassets/gammalt-innehall/rapporter/2011/guide-for-lyckad-e-handel1.pdf>
4. Connectel AB, Introduction. 2020. Accessed 2020-05-21. Available from: <https://www.connectel.co/about-connectel/>
5. Connectel AB, Vision. 2020. Accessed 2020-05-21. Available from: <https://www.connectel.co/about-connectel/>
6. Digite, What is a Kanban Board?. 2020. Accessed 2020-05-19. Available from: <https://www.digite.com/kanban/kanban-board/>
7. Black bear, Understanding Color and The Meaning of Color. 2020. Accessed 2020-05-25, Available from: <https://www.blackbeardesign.com/blog/graphic-designers/understanding-color-the-meaning-of-color/>
8. Mozilla developer, What is JavaScript? 2020. Accessed 2020-05-17. Available from: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript
9. Mozilla, CSS. 2020. Accessed 2020-05-25. Available from: <https://developer.mozilla.org/en-US/docs/Learn/CSS>
10. W3Schools, What is HTML? 2020. Accessed 2020-05-19. Available from: https://www.w3schools.com/whatis/whatis_html.asp
11. MySQL, What is MySQL? 2020. Accessed 2020-05-19. Available from: <https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html>
12. Node JS, About Node. 2020. Accessed 2020-05-17. Available from: <https://nodejs.org/en/about/>
13. Github, Axios Readme. 2020. Accessed 2020-05-19. Available from: <https://github.com/axios/axios/blob/master/README.md>
14. Github, Github. 2020. Accessed 2020-05-17. Available from: <https://github.com/>
15. Mozilla, HTTP-methods. 2020. Accessed 2020-05-25. Available from: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods>
16. Mozilla, Cookies. 2020. Accessed 2020-05-25. Available from: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies>