Faculty of Computer Science - Aalto University

Design of WWW Services D

Course Instructor: Prof. Petri Vuorimaa (2023)

Lars Oberhofer - lars.oberhofer@aalto.fi - 101876641
Nicola Casagrande - nicola.casagrande@aalto.fi - 101410544
Ferran Montoliu Boneu - ferran.montoliuboneu@aalto.fi - 101405313

December 4, 2024

Contents

1 Introduction **3**

2 Existing competitors and predecessors **3**
2.1 Attending Events . . . 3
2.2 Online purchase . . . 4
2.3 Trading . . . 4

3 User groups and use cases **4**
3.1 User groups . . . 4
3.2 User Interview . . . 5
3.3 Personas . . . 9
3.4 Scenarios . . . 10
3.4.1 Anna . . . 10
3.4.2 Bert . . . 10
3.4.3 Chris . . . 10
3.5 Use cases . . . 11

4 Content **14**
4.1 Structure of the site . . . 14
4.2 Description of each page . . . 15
4.3 Production and maintenance process . . . 16
4.4 Copyrights . . . 17
4.4.1 Ownership of Content . . . 17
4.4.2 Intellectual Property Protection . . . 17
4.4.3 Licensing . . . 17

5 User interface **18**
5.1 Description of the UI, navigation and interaction . . . 18
5.1.1 Log in . . . 21
5.1.2 Sign up . . . 21
5.1.3 Home . . . 23
5.1.4 Menu . . . 23
5.1.5 FAQ . . . 23
5.1.6 Patch details . . . 26
5.1.7 Patch details - trading . . . 26
5.1.8 Trade history . . . 29
5.1.9 Trade details . . . 29
5.1.10 Profile . . . 32
5.1.11 Add a patch . . . 32
5.2 Fonts and colours . . . 32

6 Technologies and architecture **32**
6.1 Dynamic functionalities . . . 32
6.2 Technology choice . . . 34
6.3 Architecture and structure of the database . . . 35
6.4 Security . . . 37

7 Project design **38**
7.1 Breakdown structure for the project 38
7.2 Task estimation and prioritization 38

8 Project Timeline Overview **41**
8.1 Responsibilities for each group member 43

9 Final reflection **43**
9.1 Reflections from the demo phase 43
9.2 Future work 44
9.3 Final reflections 45

10 Focus area **46**

References **47**

1 Introduction

Overalls are an important student tradition in the polytechnics of the Nordic countries such as Sweden and Finland, but also Canada. This tradition started at KTH Royal Institute of Technology in the late 60s and quickly spread out all over Sweden and Finland. Overalls are produced by student associations and have different colors based on the study tracks of the students who wear them. They are mainly used by engineering students, however they also spread out to students from other schools. Students are very proud of their overalls and wear them whenever they participate in any social activity.

The most interesting part about overalls are probably patches, patches allow students to customize their overalls and make them unique. They can be bought or obtained in many ways; for example, by participating in specific events or performing particular tasks. However, they can also be bought from associations or online to collect the most beautiful ones and show them to everyone. Another special thing about patches is that they can be traded. Many students trade their patches with other students to collect patches from other chapters/guilds and other associations that they are not part of, or to obtain patches that are not produced any more or are only produced in other universities. Patches are usually traded at specific events, where students bring along all the ones they are willing to trade and try to find other students to trade with, otherwise patches are also traded with friends or strangers on random occasions or after agreeing to the trade.

PatchMarket is a mobile-first web application that simplifies the process of trading patches by allowing students to showcase the patches they are interested in trading, and other students to offer them their patches in exchange. They can then share contacts and agree on how to trade the patch. This would make it way easier for students to find the patches they want to sew on their overalls. To allow this, the platform will require the users to register an account and login with their email and password, the system will also enable them to logout. They will then be able to upload their patches to the system by providing an image of the patch and an appropriate name, but also to add additional information if they wish to do so. Among the patches they uploaded, users will then choose the ones that they intend to trade, these patches will be seen and browsed by all the other students on their home pages. When a user finds a patch they are interested in, they can make an offer by choosing one or more of their patches to trade in exchange. Finally, users will see a history of all received and sent offers and be able to accept or decline the offers they received and cancel the offers they sent. Once an offer is accepted, the two will share their telegram accounts and start chatting to agree on how to make the trade happen.

2 Existing competitors and predecessors

There are various competitors to our proposed system. In general, students can acquire patches through different means: by attending events where they receive them, by purchasing them, or by engaging in trading.

2.1 Attending Events

Our system will not compete with the acquisition of patches through events. For the sake of simplicity and inclusivity, we have also decided not to incorporate a payment system. This ensures that everyone has an equal opportunity to trade, provided they

already possess some patches to trade or can acquire patches from their own guilds and associations.

2.2 Online purchase

Several websites offer patches for purchase, such as eBay (*eBay*, n.d.), Etsy (*Etsy*, n.d.), or other less commercial platforms. The well known platforms have the advantage of an already strong established brand presence. They also offer shipping to the northern European countries and have a competitive pricing. The less commercial platforms are private collectors' website "Patch Collection" (*Patch-Collection*, n.d.) and "The Carolina Trader" (*The Carolina Trader*, n.d.). However, these platforms require payment. Our platform aims to encourage students to exchange Telegram usernames and discuss meeting places to facilitate agreed-upon patch trades. The downsides however are that the shipping costs can be quite high, it might take days up to weeks to receive and the payment obligation which is not what we wanted to indulge with.

2.3 Trading

In-person patch trading typically occurs at events where patch trading is a central theme, such as jamboree events organized by scout associations (*Scout Jamboree events*, n.d.). However, this approach has limitations in terms of flexibility, as it is restricted to specific dates and locations. Meeting in person, such as on campus, is the simplest method, but students often face difficulties finding other students interested in trading. This is a problem that has yet to be addressed, and it's what we aim to resolve with PatchMarket. Online trading is hard to find and seems cumbersome. On Reddit (*Reddit Trade Patches thread*, n.d.) there are sub-reedits which offers community based patch trading. It is a private community and one needs to be verified to access it. Also, the target group is more in the United States instead of our user group.

3 User groups and use cases

First, the user groups are defined. To define personas, user interviews are then conducted in the respective user groups. Afterwards, scenarios can be derived from the personas, which can then also be found in the use cases as a process.

3.1 User groups

As described before, PatchMarket aims to serve students in northern European countries such as Finland and Sweden. The main user group for our web service, therefore, consists of students in northern European countries where trading overall patches is a well-known practice. This user group can be further divided into the following categories:

1. Students from Finland and Sweden
    - These students make up the core user group, as they are familiar with the concept of collecting patches and are most likely to use our service.
    - They also typically have access to patches from their study guilds or associations.

2. Freshman Students
    - New students who do not have many patches for their potential overall just yet and could be more eager to collect patches, hence using our service.
3. Active Guild Students
    - These students are more eager to represent their guild and exchange guild patches.
4. Students Who Attend Many Events
    - The more events students attend, the more patches they accumulate, making them more likely to want to showcase their patches to friends through our service or trade them for better ones.

Dividing the user groups into these groups helps us to understand the different needs and preferences of our target group and to adapt our system accordingly.

3.2 User Interview

To better understand our user group, their goals and their needs, we conducted user interviews. From the interviews we found several pain points and made discoveries.

In the first interview (Figure 1) we learned that shipping patches when trading them might be a hurdle for users since the shipping costs could be significant. We also learned that it is common to trade your own guild or association patch, since these are the ones that students have easier access to and can easily buy multiple times. In addition, the interviewee mentioned that people do not swap as much in person, as it is rare to find patches that match one's taste.

In the second interview (Figure 2), the participant confirmed to us that the most commonly traded patches are the ones from guilds and associations. Patches from events instead are less traded since they usually indicate that a person has been to that specific event. However, the interviewee mentioned that collecting patches was for him more interesting in the beginning of his studies. Once you have enough patches, there seems to be less need to collect more or better ones.

In the third interview (Figure 3), we diversified our demographics by interviewing an international student studying at Aalto University. One of his pain points was that he was not able to pay for patches during his previous studies in Sweden, since he did not have Swish nor cash with him most of the times. The situation for him is similar also in Finland, since he does not have Mobile Pay yet, and thus cannot pay for patches if he does not have cash. The interviewee mentioned that he would also trade if there were nice patches to swap.

These key takeaways are used to validating our user group, creating personas, and providing more concrete ideas for the functionalities in our prototype.



Figure 1: User interview - Interviewee 1

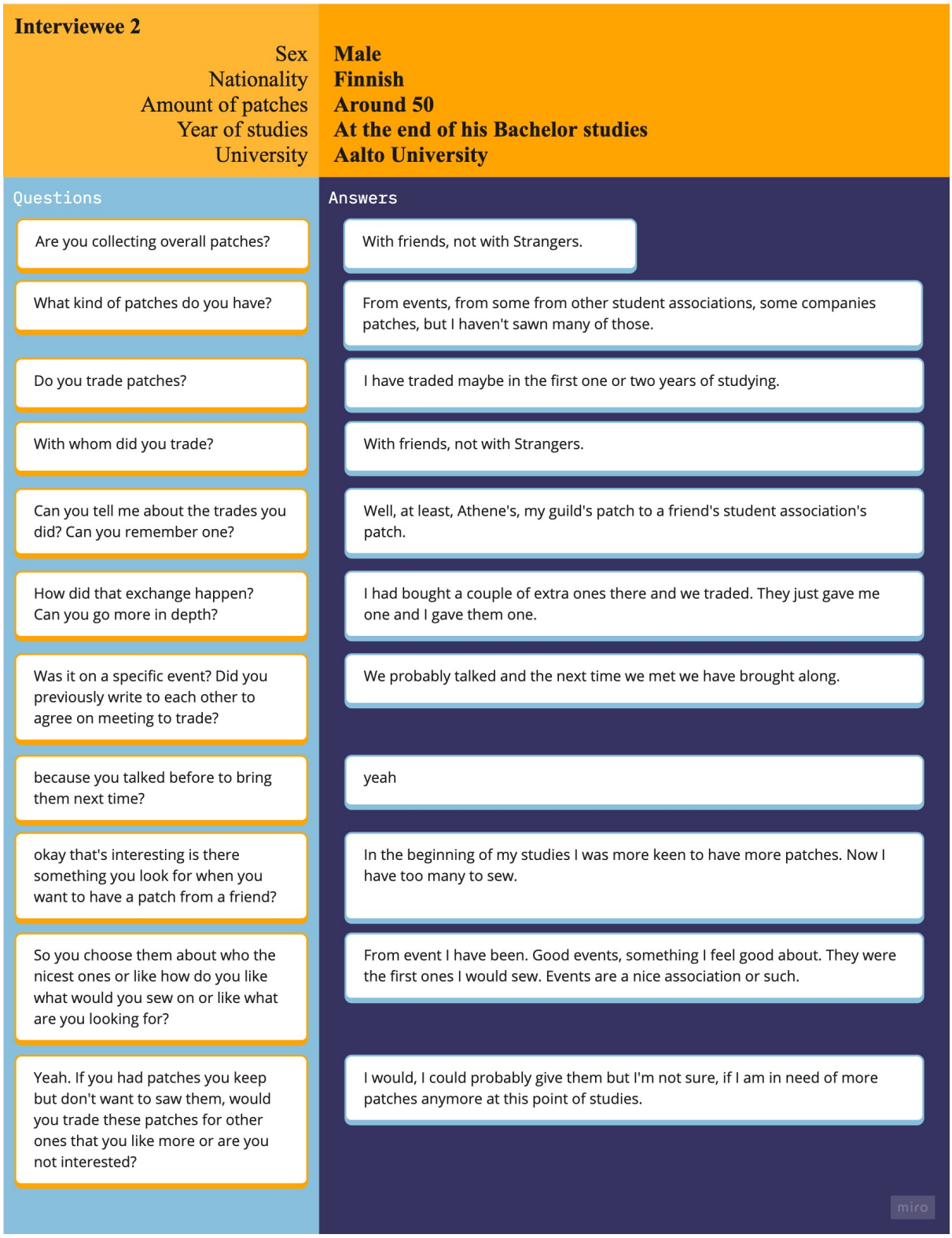


Figure 2: User interview - Interviewee 2

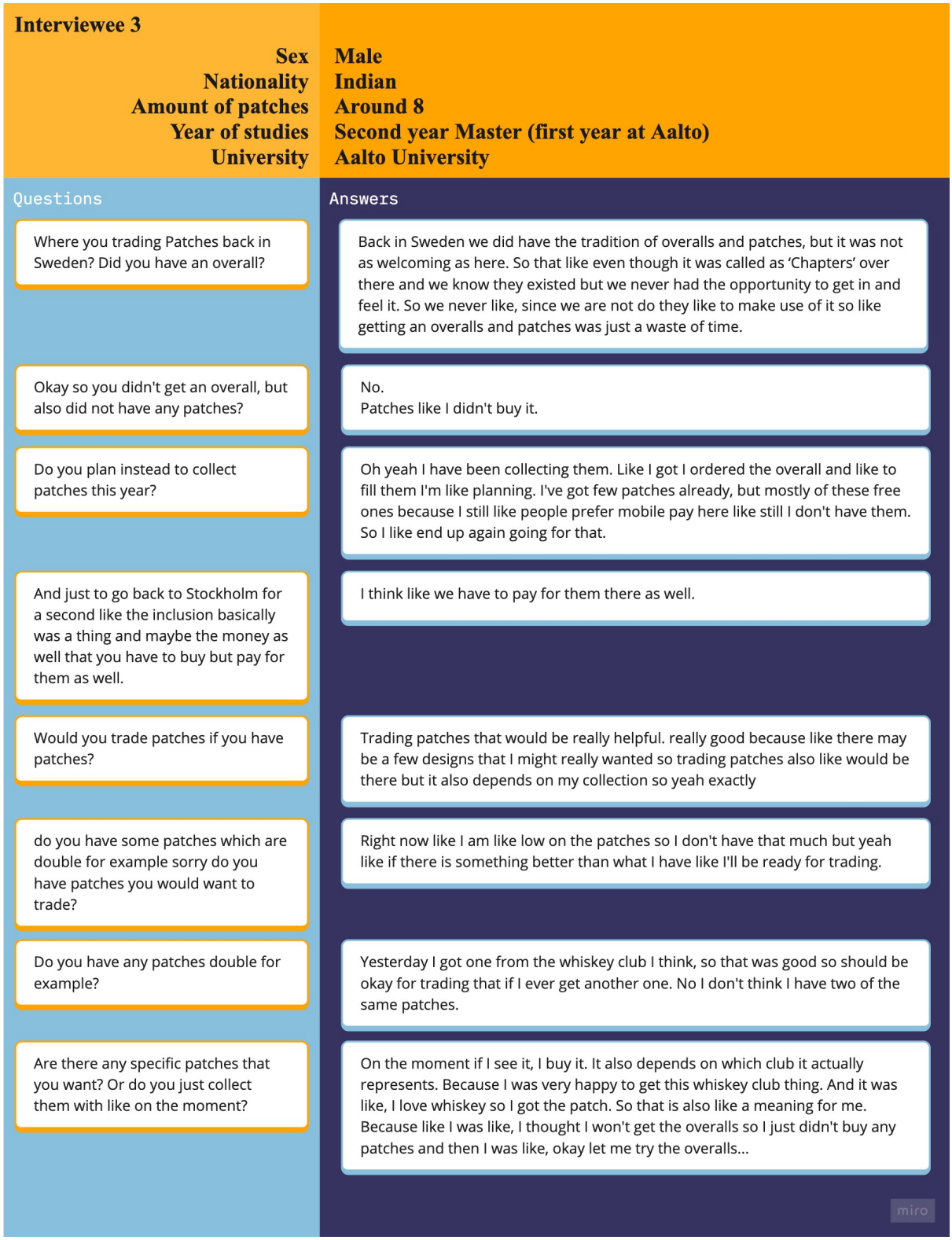


Figure 3: User interview - Interviewee 3

3.3 Personas

After the user group is defined and story interviews conducted, we can use the gained insights to establish personas. Two normal users and one extreme user are considered. The advantage of an extreme user is that it challenges our design and makes us look at it from a different perspective. These personas help us to understand our design better.

Persona 1	Name	Anna				
Characteristics	Age	19	Gender	Female	Job	Bsc. Computer Science
	Background new student from Italy, active member of guild association Characteristics: limited budget, tech savvy, collects patches as hobby, eager to expand her collection Pain points: shipping costs are a hurdle to trade patches, she only has access her own guild patches. she rarely finds patches she likes when trading in person. Goal: Anna wants to have a full overall by then of the school year and plans therefore to collect 30-40 patches, she is considering trading two patches for one and trading her association patches for others to achieve her goal.					

Persona 1	Name	Ben				
Characteristics	Age	21	Gender	male	Job	HCID bsc. student
	Background: found collecting patches fascinating but has slowed down as his collection has grown. Characteristics: collected lots of patches from events, little motivation to actively seek new patches, prefers trading guild patches. Pain points: there are a few patches he would like but cannot find, difficult to find patches with a meaning. Would like to show his collection to friends. Goal: Ben wants to show his patch collection through the application, making it easier to share them with friends and he hopes to use the app to find his last missing patches.					

Persona 1	Name	Chris				
Characteristics	Age	25	Gender	male	Job	Business master student
	Background: international Erasmus student and not a typical northern Europe student, new to the concept of patches and overalls, Characteristics: would like attractive patches, he is very enthusiastic about it Pain points: he cannot buy patches easily (does not have mobile pay) and he does not belong to any guild or association to get them. Goal: Chris plans to trade patches he acquires through his association on Patchmarket. He currently faces difficulties buying patches at events due to varying payment systems,					

Figure 4: Personas: Anna, Bert and Chris

All three personas fall within our user group. Anna is a freshman eager to collect many patches. Then there is Bert, a more experienced student, who is still looking for a

few additional patches. Chris is our extreme user who has not yet decided to join a guild.

3.4 Scenarios

3.4.1 Anna

Anna, a new Italian student at KTH in Sweden. She has already joined her study guild and the tennis association. She wants to have a full overall by the end of the year. She tried to buy patches online, however, her tight budget and shipping costs make buying patches a challenge. When she talks about patches with people they often show her some of their collected patches and the ones they offer to trade, however she rarely finds patches she likes. Anna discovers our platform through one of our posters, since she is always curious and sees a poster about PatchMarket and searches it up and signs up. She is happy because her dream of having an overall full of stickers became much easier and within reach. She clicks on list patches for trading and uploads 10 patches of her Tennis association and 10 patches of her study guild.

3.4.2 Bert

Bert used to collect patches passionately, but during his last year he barely continued his collection. His collection, primarily from events and guilds, has left him with a few patches he still would like to have. He has seen them on other people's overalls and would also like to get them.

He values patches with a meaning, and he likes showing his collection on his overall to his friends. He stumbles on our platform and finds a community of patch enthusiasts offering all kinds of patches. He creates an offer for one of the patches he would like. On the next day, he checks the website and sees that user Ivan has accepted his offer. He can now see his Telegram name and sends him a message.

3.4.3 Chris

Chris, an international exchange student from France, starts his minor at Aalto University in Finland. He is warmly welcomed and decides that he also would want to collect patches. He saw some stand during the orientation week, but he did not have cash money and mobile pay does not work with his French bank account.

He finds this website called Patchmarket his Finnish friends tell him about. He looks it up and lands on the landing page of the website. Since he has not created an account yet, he chooses to register. He provides his name, email address, his telegram name and chooses a password. Afterwards, he starts searching for patches. He finds a patch he likes a lot and clicks on it to see more patch details. He clicks on sending an exchange offer, however he quickly realizes that he cannot trade patches since he does not have any patches to trade with.

One week later, he has some very lovely conversations with the guild members from Athene. They even let him try on different overalls to match his potential size. He realized that he would want to have an overall so that he can sew the patches on.

More importantly, he can buy cheaply Athene guild patches and then trade them only with other patches he fancies.

3.5 Use cases

This section illustrates the use cases of the application. To give a clearer view, we divided the use cases into three categories, and we present them separately. Each of the use case diagrams is followed by a description that illustrates all the specific use cases that are presented in the diagram.

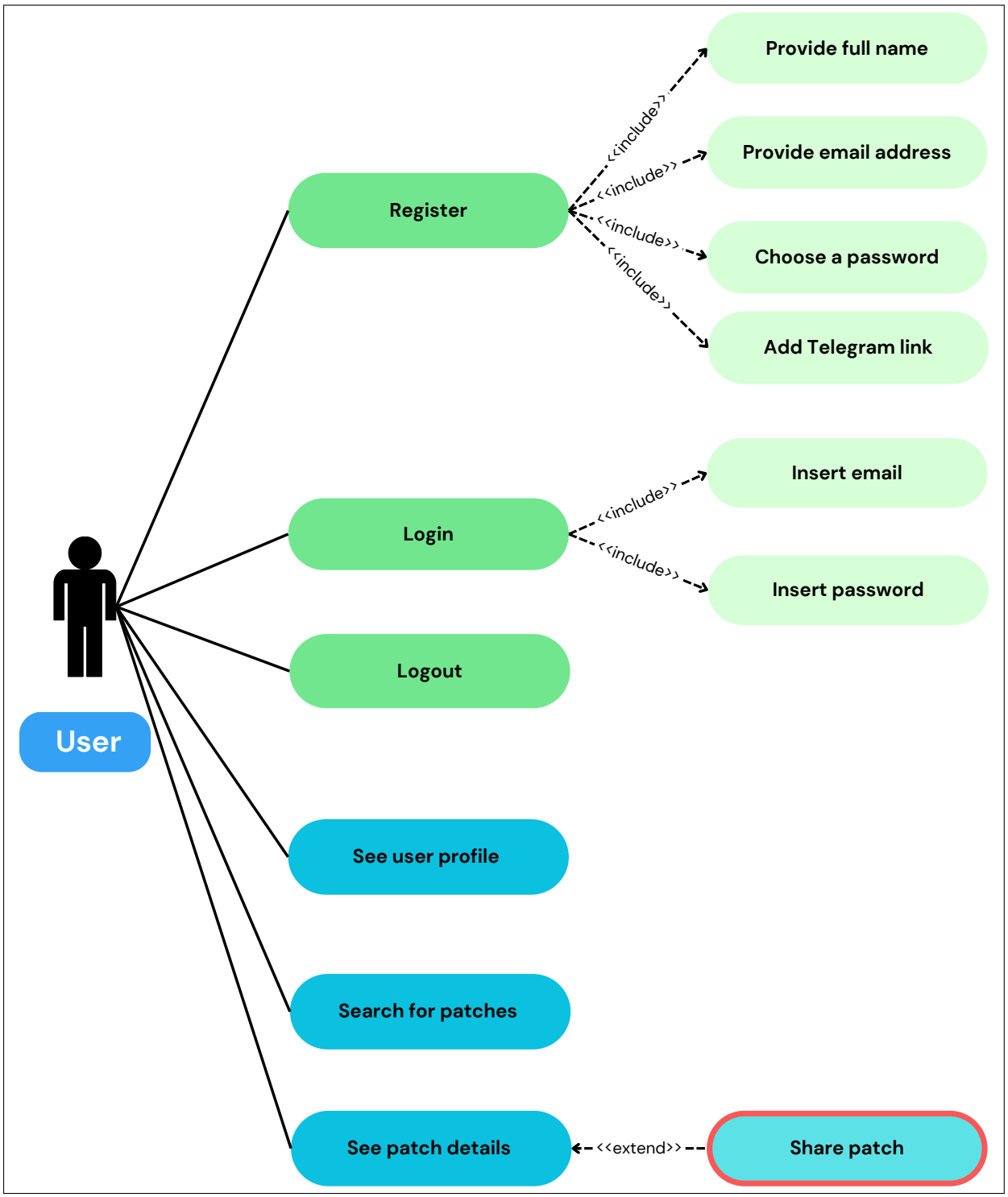


Figure 5: Authentication, User Profile and Search - Use cases diagram

Figure 5 presents the use cases: *registration*, *login* and *logout*, which all users need in

order to utilize the service. Furthermore, they are also able to visualize the information on their profile. The use cases: *search* and *see patch details* are also available to all users of the platform and do not require uploading any information on the website. More in detail, these use cases are:

- **Registration:** create an account on the application by registering with their full name and email and by choosing their password.
- **Login:** login to the application using email and password to be able to use the service.
- **Logout:** logout from the application.
- **See user profile:** visualize the information on their user profile: including their full name, email address, telegram link and patch collection.
- **Search for patches:** browse the patches that are listed for trading, with the possibility to search for them by name.
- **See patch details:** visualize the details of a patch (name, picture, university, description, categories). Share a patch by copying its link.

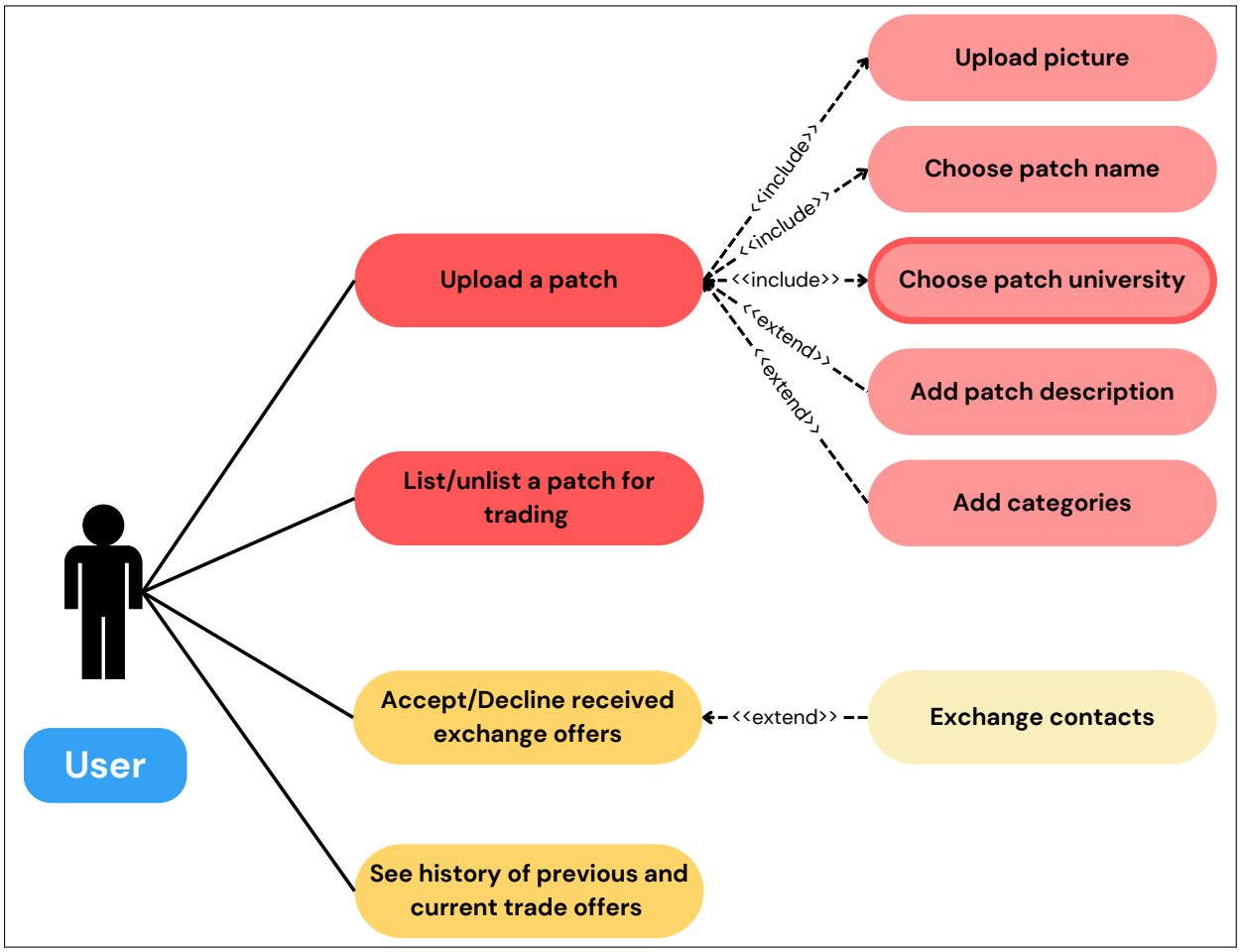


Figure 6: Listing a patch and receiving exchange offers - Use cases diagram

Figure 6 presents the use cases: *upload patch* and *list/unlist a patch for trading,* which are required by users that intend to list a patch and receive offers from other users on the platform. The use cases *accept/decline received offers* and *see history of previous and current trade offers* are instead needed for the users to answer to the received offers and manage their trades. More in detail these use cases are:

- **Upload Patches to their collection:** upload a patch by providing a name for the patch, a picture, the university of origin, an optional description and an optional list of categories that describe the patch.
- **List/unlist a patch for trading:** choose a patch from their collection and list it for trading to let other people know about the fact that they want to trade the patch and enable them to make offers. Unlist the patch if they change their mind and no longer want to receive offers for that patch.
- **Accept/Decline received exchange offers:** choose whether to accept the received offer and trade the patch, or refuse it and renounce to the trade.
- **See history of previous and current trade offers:** see the history of all the offers they sent or received on the website and all the transactions that are still pending.
- **Exchange contacts with the other person:** give the other person your contacts to finalize the trade and agree on the place where to meet or the shipping options.

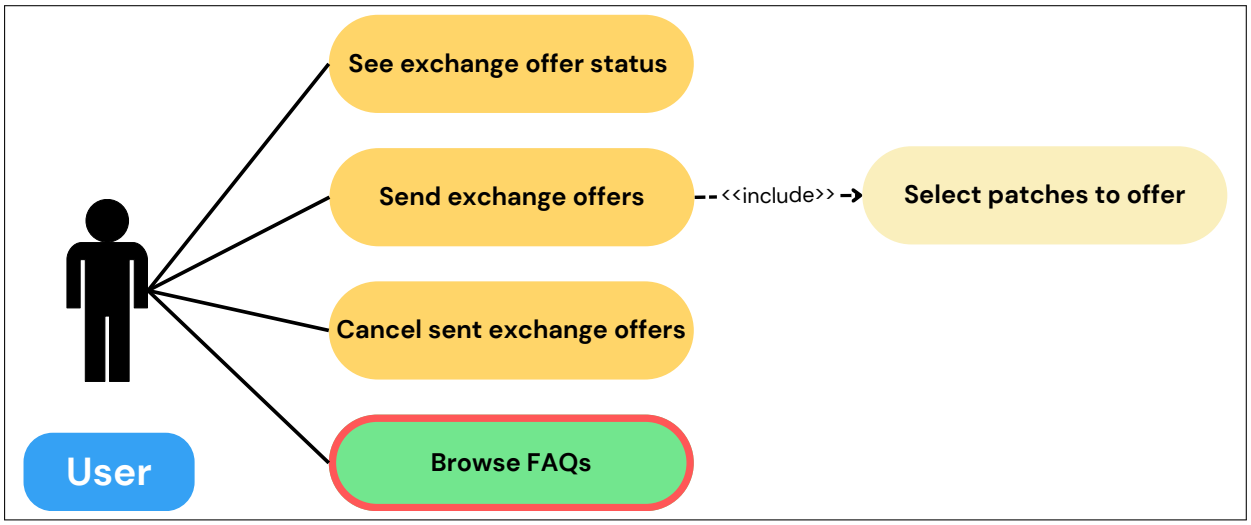


Figure 7: Make an exchange offer - Use cases diagram

Figure 7 presents the use cases: *send exchange offer*, *see exchange offer status*, *cancel sent exchange offers* which are required by users that want to send exchange offers to others, and enable them to cancel such offers in case of need.

Moreover, it introduces the use case *Browse FAQs* which allows users to browse the answers to commonly asked questions about the platform and trading their patches. More in detail these use cases are:

- **Send exchange offers:** send exchange offers to other users by selecting patches from their collection.

- **See exchange offer status:** see the status of exchange offers, which can be:
  - **Pending:** the offer did not receive any answer yet.
  - **Accepted:** the other person accepted the trade offer.
  - **Declined**: the other person refused the trade offer.
  - **Cancelled:** the person who sent the offer cancelled the offer.
- **Cancel sent exchange offers:** cancel an exchange offer to let the other person know that it is not valid anymore.
- **Browse FAQs:** browse frequently asked questions to find information about using PatchMarket and trading patches on the platform.

4 Content

4.1 Structure of the site

The site is a Single Page Application (SPA), which means that the user is always in the same page technically, but the contents of said page update through state changes. In the flow diagram from Figure 8, a rounded square represents each screen or page that we can have in our application. For the sake of simplicity, we will refer to said screens or pages with this name instead of calling them states or widgets.

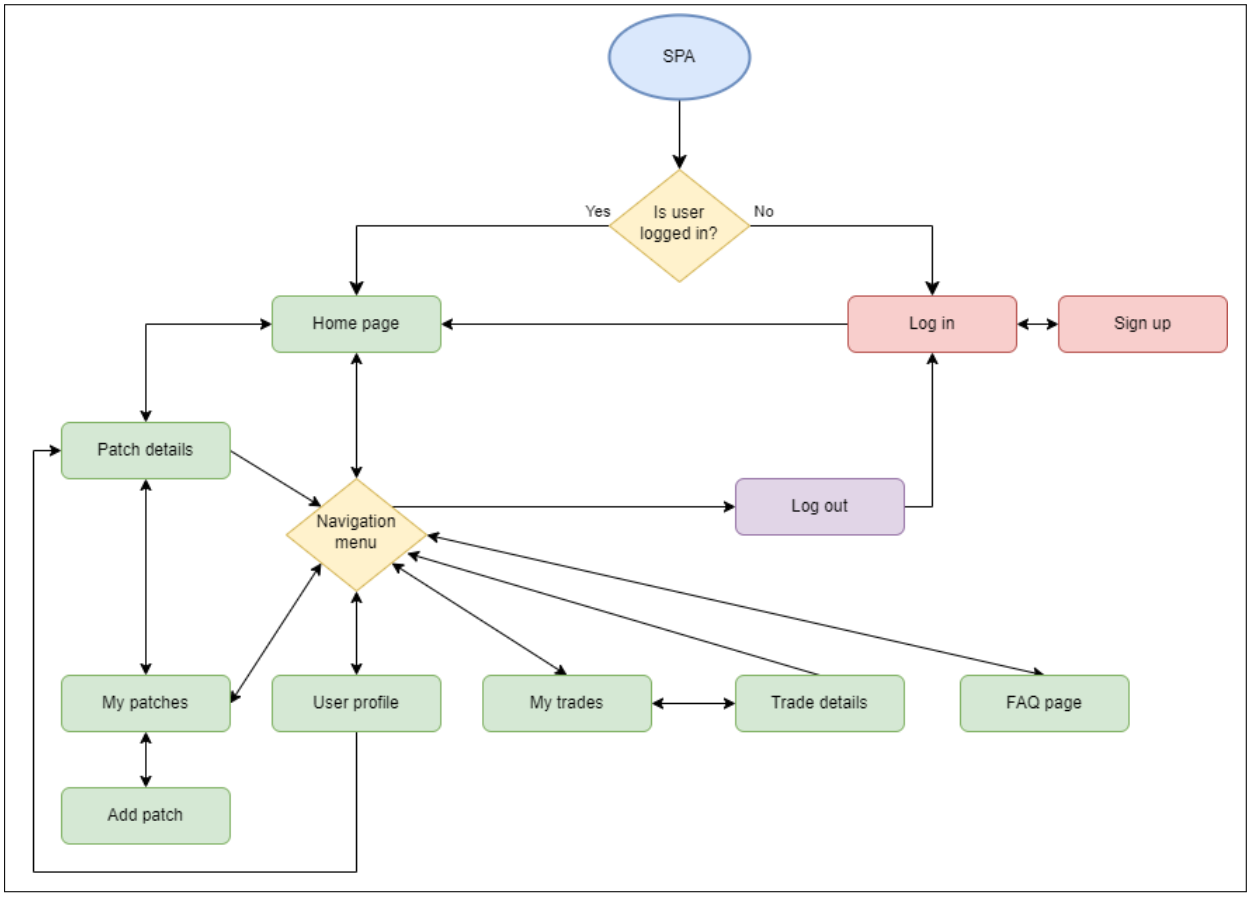


Figure 8: Site structure - flow diagram

In the diagram from Figure 8, the first element is the SPA. This means to represent that the user loads the page, so it is the starting point of the diagram.

When a user opens the platform, the first thing that the system checks is if the user is logged in or not. If they are not logged in, then they are redirected to the login page, from which they can only navigate between the sign-up and the login pages. The screens available to a user that is not logged in are represented in red, while the ones that are available to logged-in users are represented in green.

If the user logs in (or if they were already logged in), then they are redirected to the home page, from which they can access a navigation menu that allows them to go to any of the following pages:

1. **Home page:** from there, a user can navigate to the details page of any patch.
2. **My patches:** from there, a user can navigate to the details page of any patch. A user can also navigate to another page which allows them to add a new patch.
3. **My trades:** from there, a user can navigate to the details page of any trade offer.
4. **User profile:** from there, a user can navigate to the details page of any patch that they own.
5. **FAQ:** this is a page that explains how the platform works. From there, the user cannot navigate to any other page (except by using the navigation menu, of course).
6. **Log out:** from there, the user is automatically redirected to the login page. That is why this is not represented in red nor green, since it's actually not a page, but a step used to perform a quick action (log out the user and invalidate their session), and then redirect them to the login screen.

In the diagram one can also see that any of the screens that were represented in green (the ones a logged-in user can access), can interact with the navigation menu, which allows the user to navigate to whichever of the other screens they want (except the details pages, which can only be accessed through specific pages).

Then, a normal flow for a user would be to first log in, then get redirected to the home page and then, from there, use the app interacting mainly with the navigation menu to switch between all the different functionalities. Whenever the user wants to close the session, they can use the navigation menu to access the log-out screen, which will automatically redirect them to the login screen, after logging them out.

4.2 Description of each page

This section briefly describes each of the pages of the application and the functionalities that they offer. Specifically, PatchMarket includes the following pages:

- **Log in:** this page is the first page that is shown to the users if they are not already logged in. It allows them to log in using their email and password, if they already have an account, and to navigate to the home page. It also allows them to reach the sign-up page in case they wish to create a new account.
- **Sign up:** this page can only be reached from the login page and allows users to create a new account on the platform by providing their full name, email address, telegram username and an appropriate password.

- **Home page:** this page is the main page of the website and allows users to search for patches by name and to filter them by categories, but also to browse the results. From the home page, it is possible to navigate to the patch details page of a specific patch to check out its details and make an exchange offer.
- **Patch details:** this page allows users to see the details of their patches, and of patches that are offered by other users. It also allows making trade offers for the patches that are listed for trading by others.
- **My patches:** this page allows users to visualize a list of their patches and to navigate to the patch details page to see their details. It also enables them to navigate to another page to add new patches to their collection.
- **Add patch:** this page allows users to add a new patch to their list of patches through a simple form.
- **My trades:** this page allows users to see a history of all the trade offers they sent and received in the past and the pending offers. By clicking on one of the offers, they can navigate to the trade details page.
- **Trade details:** this page allows users to see the details of a specific offer by visualizing the patches they would give and the ones they would receive. They would then be able to accept or decline the offer in case they received it, and to cancel it in case they sent it.
- **User profile:** this page allows users to visualize the details of their profile and a preview of the list of their patches.
- **FAQ:** this page is used to explain how the platform works overall, and to provide answers to the most frequently asked questions. These questions were based on what we think may not be clear enough, and on the feedback we got from potential users during the Demo gala.

4.3 Production and maintenance process

For the production of the service, the process is the following:

First, we gathered enough data about our user group through some interviews. With this data, we came up with a mock-up which, after some iterations, we turned into a high fidelity prototype.

After the design was complete, we designed the database based on the requirements and then proceeded with the development. During the development, we made a complete regression test each time we deployed a new version of our service. This means that, whenever one person implements/updates anything, they are responsible to test said functionality and make sure that nothing breaks.

For the deployment, we use a platform called render, which you can link to a GitHub repository and will update the deployment each time a commit or pull request is made to the main branch. The database is a remote database which is hosted in MongoDB Cloud, so it is persistent between deployments.

Right now, we are facing some performance issues due to render, the platform we used to deploy our service. That's because, since we use the free plan, if the application has not

had any requests in the last 15 minutes, they automatically shut down the deployment and then, whenever a user makes a request after that time, it has to build and deploy things again, which takes around 30 seconds. This can easily be fixed by using the paid version, but for the scope of this project and the current state of the platform, we decided to keep the free version for now.

Regarding the maintenance process, thanks to the technologies and architecture we use, the maintenance needed is very low. We expect to have very few to no runtime errors, and to make sure of that we use TypeScript and eslint for the development. This reduces the maintenance process by a lot, since we don't expect to invest much time in fixing errors in the production environment. Nevertheless, the app still needs some maintenance. For instance, we need to regularly update the dependencies (at least once a month) to make sure that we have the latest stable version, mainly for security reasons, but also to have the latest features available.

Regarding long-term maintenance, if the app were to scale to a level which required optimization, the first measures that we would do would be to check the code for potential code optimizations, migrate from Node.js to Bun, and migrate from MongoDB to PostgreSQL, as Bun and PostgreSQL are much more performant than Node.js and MongoDB. Also, when migrating to PostgreSQL, to further decrease the network delay, we would have to set up a database in the same server where the API is deployed, so that the queries don't have extra network delay. Nevertheless, we expect those optimizations to not be needed for the short and mid terms.

4.4 Copyrights

4.4.1 Ownership of Content

We do not own the images or logo used in creating the web service. The content uploaded by users who submit their images remains their property. PatchMarket holds a license to display, modify, and distribute their content within the platform. We do not claim ownership of user-generated content. Users have the right to remove their content from the platform at any time. Before publishing TradeMarket, we will comply with the Directive 2000/31 (*Copyright Directive*, 2000) Europe provided framework on Copyright.

4.4.2 Intellectual Property Protection

Before hosting the platform, we will protect branding elements associated with our web service. We will use copyright protection where applicable and display copyright notices as required.

4.4.3 Licensing

All third-party resources i.e. fonts and code libraries are subject to various licenses including open-source licenses, and commercial licenses. We will adhere to the terms and conditions specified by these licenses.

5 User interface

5.1 Description of the UI, navigation and interaction

This section introduces the final user interface of PatchMarket. The application has been developed by taking into consideration the needs of our personas, which are based on the interviews and user research that we conducted. During the design process, we also took into strong considerations industry standards and user experience principles.

To reach the final user interface design that we will present below, we started with low fidelity prototyping. We used an iPad to design the first sketches of the screens. This process allowed us to refine the design by going through multiple iterations and to take important decisions on various user experience aspects while we were still able to easily change the design.

Afterwards, we started to design high fidelity prototypes in Figma to give a clear idea of how our application will look and feel and of how its interactive functionalities will work.

Finally, we started designing the final user interface of PatchMarket using React and Mantine. In this section, we present the user interface design of both the mobile and desktop versions of the website. PatchMarket has a responsive design and looks consistent to the user in both version providing the same functionalities and a very similar user experience.

PatchMarket aims to be quick and easy to use. That is why we kept its set of functionalities to the core and designed the user interface in such a way that it allows users to quickly complete any task from any screen with just a few clicks. The navigation system is also simple and is provided in a standard way to keep it consistent with other applications the user may have used.

To prevent user errors the application always indicates to users which information is required and which is optional, it also does not allow them to proceed in case they do not complete the required fields. Furthermore, it always asks for the user confirmation through pop-ups when important operations are to be executed.

Finally, PatchMarket always provides users with feedback through notifications that confirm to them whether the operations they executed were successful or not so that they do not have to worry about their outcomes.

In the following part of this section we present all the screens of our final design and we describe each of the pages and their interactive functionalities. We also explain in details our design choices and the reasons why such choices have been implemented.

All the screens of the application include a top navigation bar that contains the PatchMarket logo and name, and a hamburger menu that can be opened and closed with a click to have access to all the sections of the application. On bigger screens, the top navigation bar directly includes the links to all the pages instead of the hamburger menu. Additionally each screen includes a footer with the logo and name of the application. Exceptions to this are the log in and sign up pages which do not contain the hamburger menu and the footer since the users are not able to access any other page if they are not logged in.

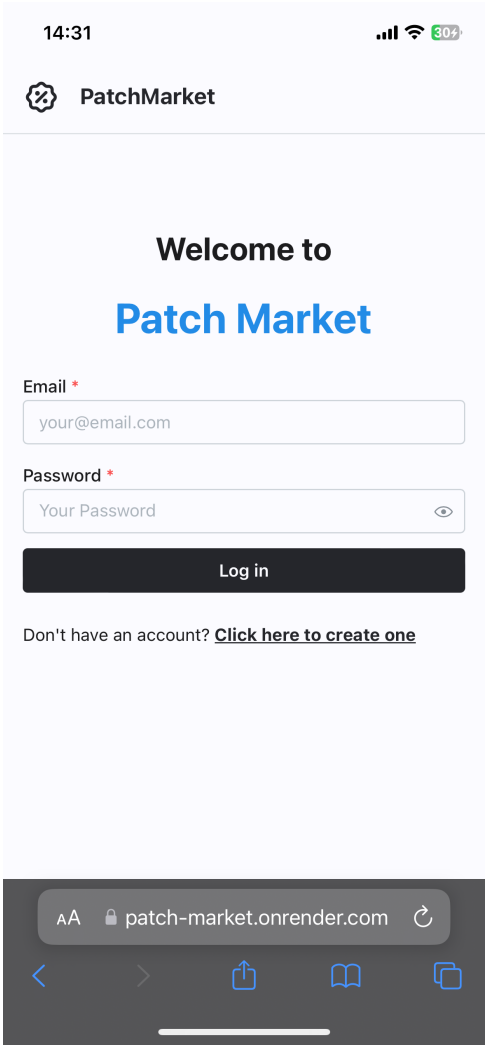


Figure 9: Log in

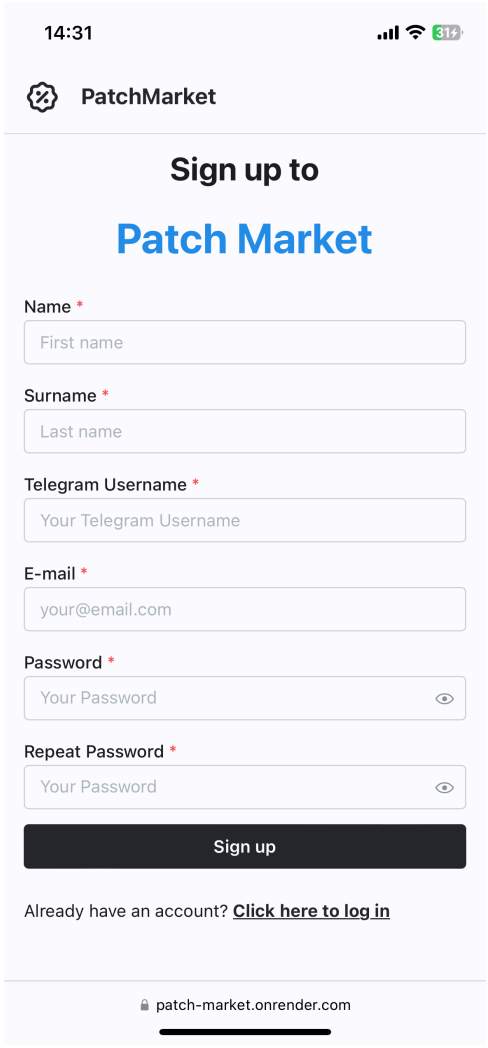


Figure 10: Sign up

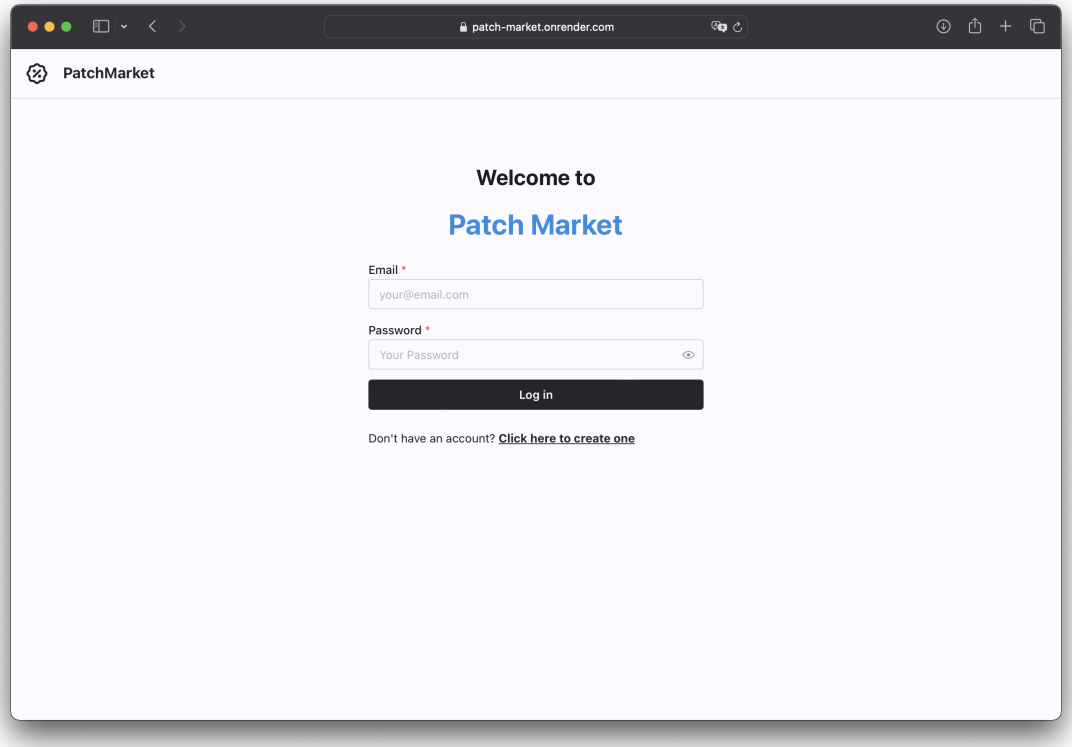


Figure 11: Log in - Desktop

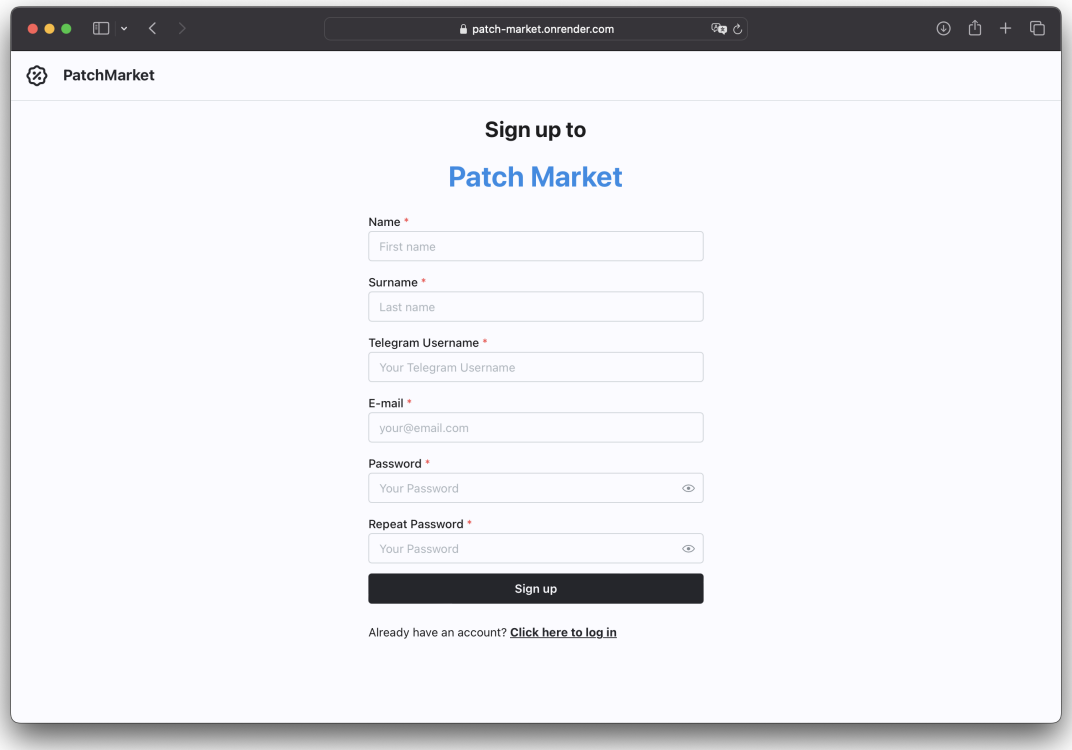


Figure 12: Sign up - Desktop

5.1.1 Log in

Figure 9 and Figure 11 represent PatchMarket's log in page. The page includes two text fields where users are required to insert their email and password. Since the two fields are required they are indicated with a red asterisk. This design follows industry standards and ensures that users are aware of the fact that the two fields are required. After inserting the requested data, users can click on the log in button to log in and be redirected to the home page. If the inserted data does not pass the checks that PatchMarket performs on the input fields, the users will see an explanatory error message under the fields where they made a mistake, this allows them to quickly understand what is wrong and insert the correct information. Moreover, if the log in process fails, they will receive a notification about what went wrong. This to give them a clear indication of what they should do to correctly log in. Under the log in page, we also include a link to the sign up page, so that users can sign up if they do not already have an account to log in with.

5.1.2 Sign up

Figure 10 and Figure 12 illustrate PatchMarket's sign up page. The page can be reached by clicking on the apposite button on the log in page and allows users to create a new account by inserting their data in the available text fields. The page contains one text field for each information that is required including: *name*, *surname*, *telegram username*, *email*, *password* and *repeat password*. All the text fields are marked with a red asterisk to indicate that they are required. If the user makes any error, they are informed in a consistent way with the login page. The repeat password field makes sure that users do not make a mistake while writing their password and register with a wrong password. Once they completed all the text fields, users can sign up by clicking the *sign up* button. The page also contains a link to the log in page to allow users to log in, in case they already have an account.

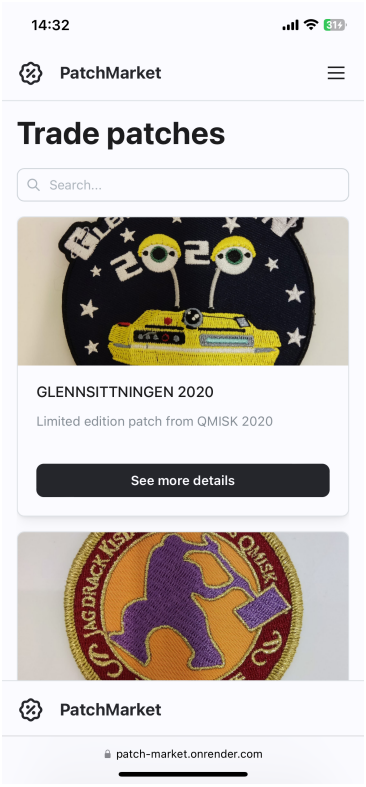


Figure 13: Home

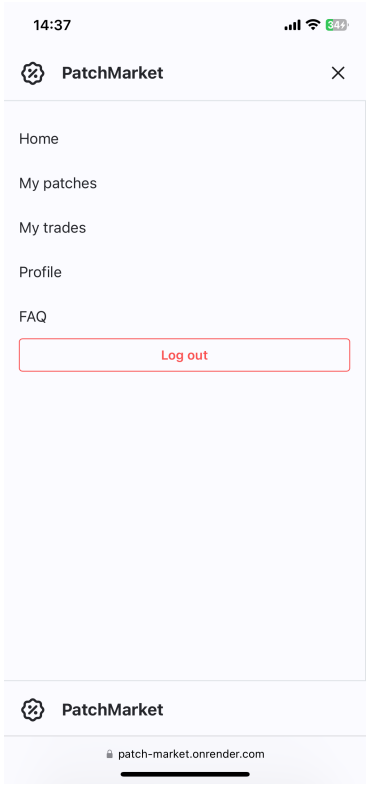


Figure 14: Menu

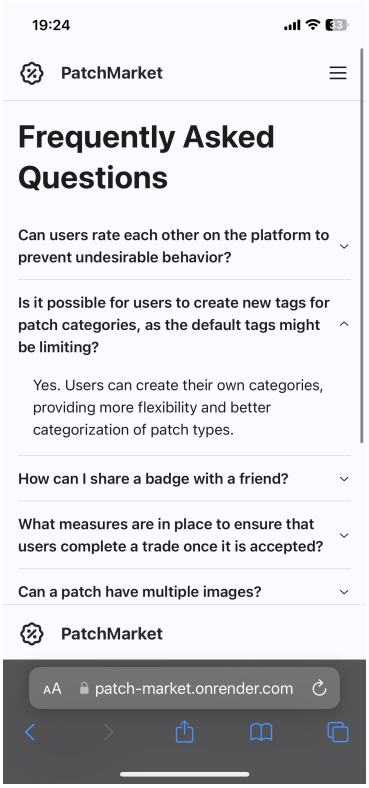


Figure 15: FAQ

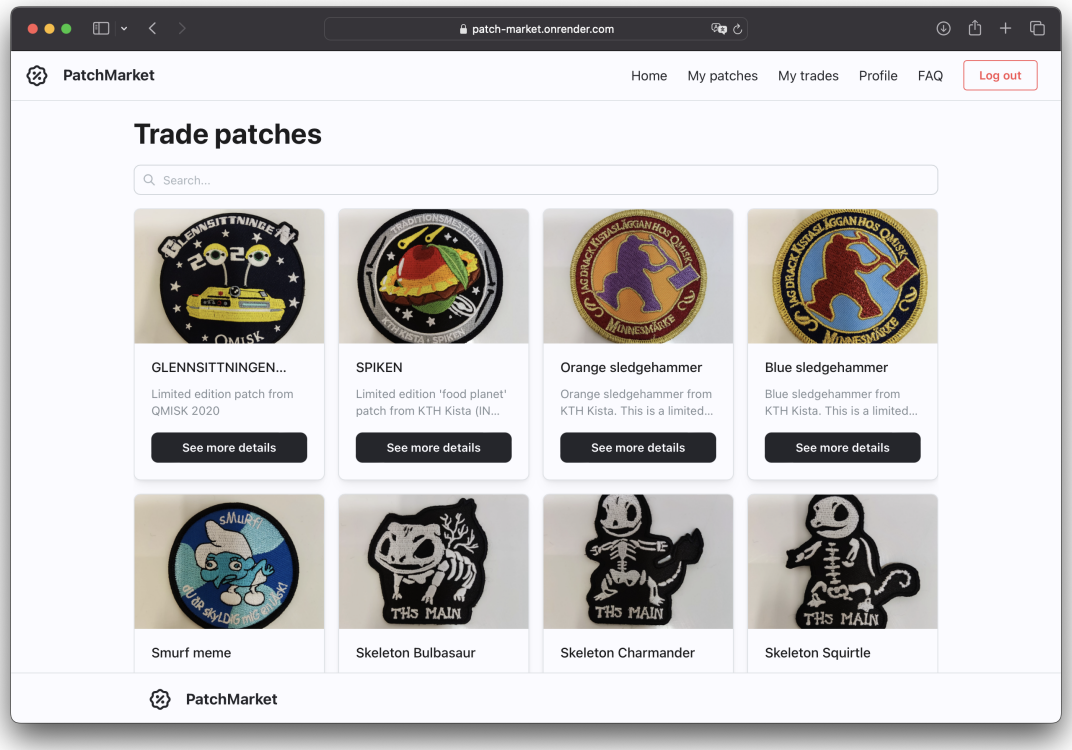


Figure 16: Home - Desktop

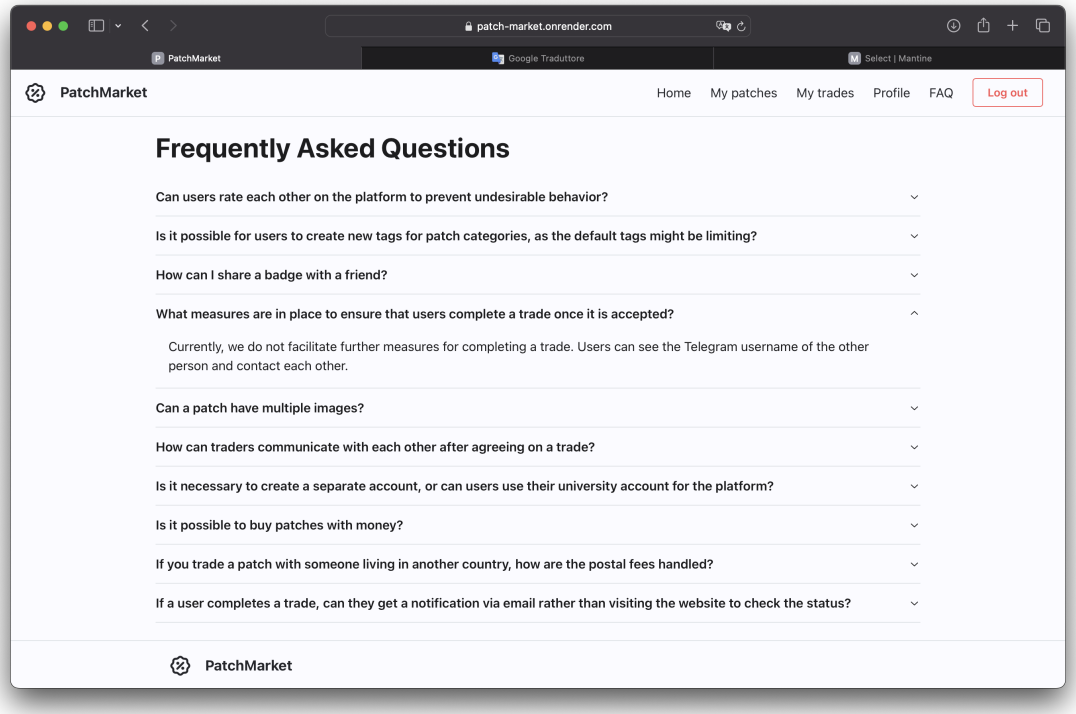


Figure 17: FAQ - Desktop

5.1.3 Home

Figure 13 and Figure 16 represent a screenshot of PatchMarket's home page both in the mobile and desktop versions. The home page allows users to visualize a list of the patches that other traders listed for trading on the platform. Each patch that is listed includes its name, description, picture and a button: *See patch details*. The button can be clicked to see the details of the chosen patch. At the top of the page a search bar is available for users to search for patches with a specific name.

5.1.4 Menu

Figure 14 presents the menu of the application that can be opened by clicking the hamburger in the navigation bar on mobile. On desktop screens, instead, the menu is directly available at the top of the home page as it can be seen in Figure 16 The menu allows to reach most of PatchMarket's pages including: *Home*, *My patches*, *My trades*, *Profile* and *FAQ*. It also allows users to log out. On mobile, the menu can easily be closed by clicking on the x button that appears in the navigation bar, or by selecting one of the options.

5.1.5 FAQ

The FAQ page is shown in Figure 15 and Figure 17. This page was added to PatchMarket during the development phase to answer the questions that our users could have about using the platform. On this page users can find questions that are commonly asked about trading patches on PatchMarket and expand them to visualize their answer. In this way they can get a better understanding on how the platform works and have their doubts solved before using it.



Figure 18: Patch Details

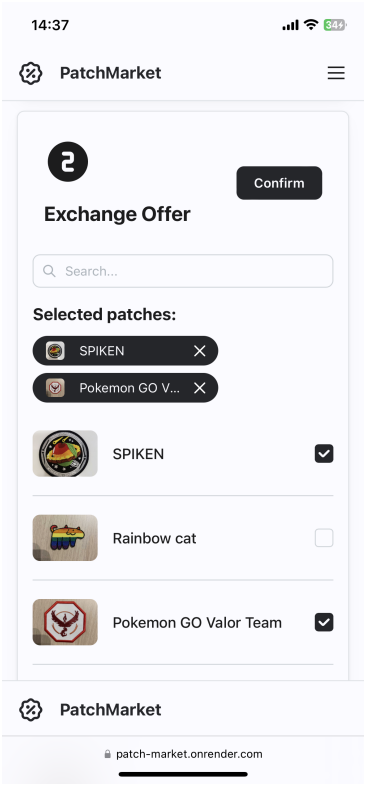


Figure 19: Trading

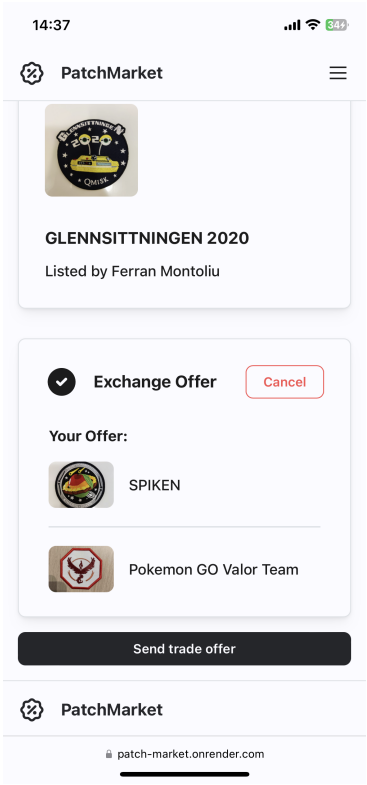


Figure 20: Offered Trade

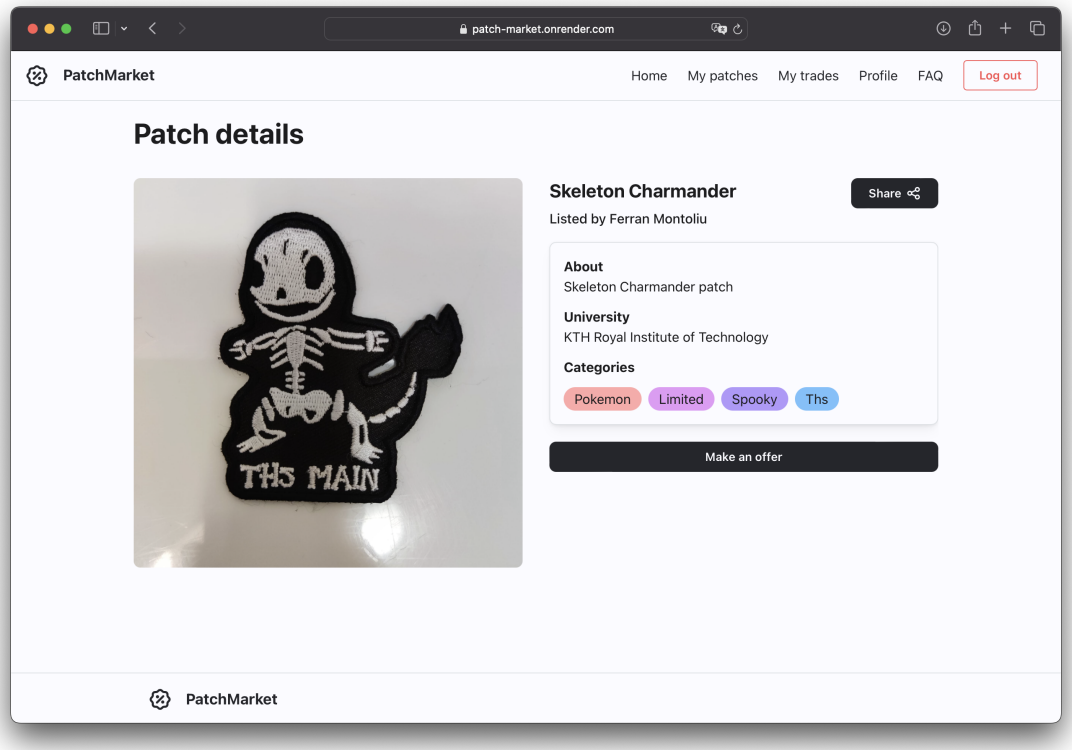


Figure 21: Patch Details - Desktop

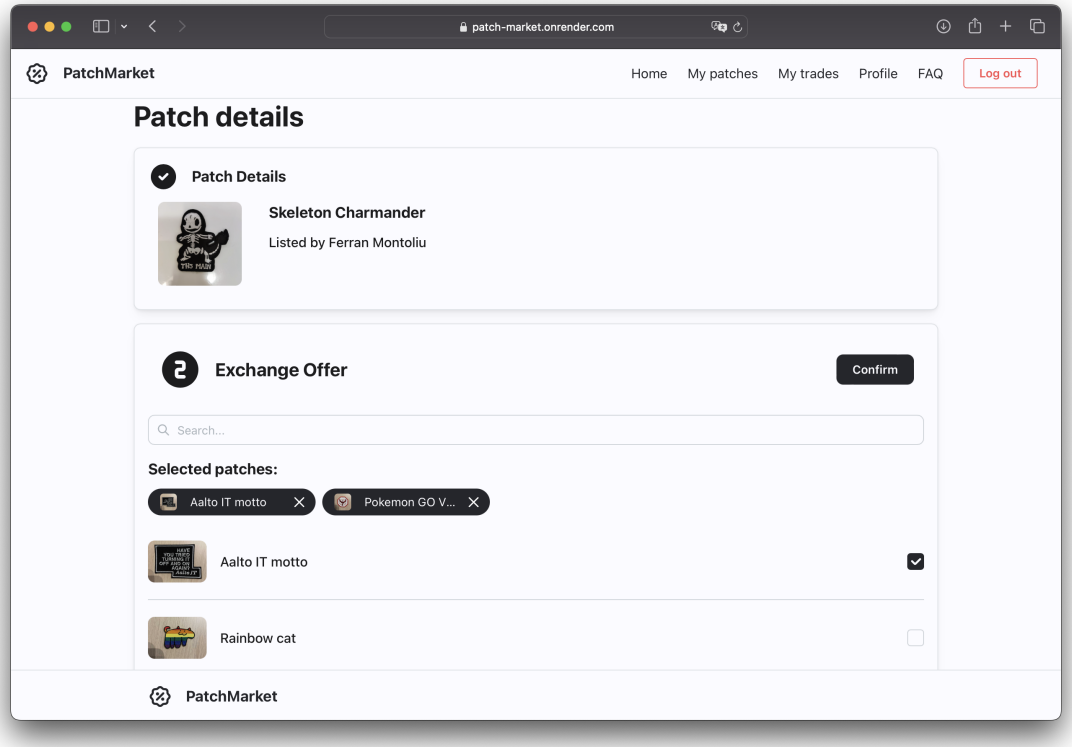


Figure 22: Patch Details Trading - Desktop

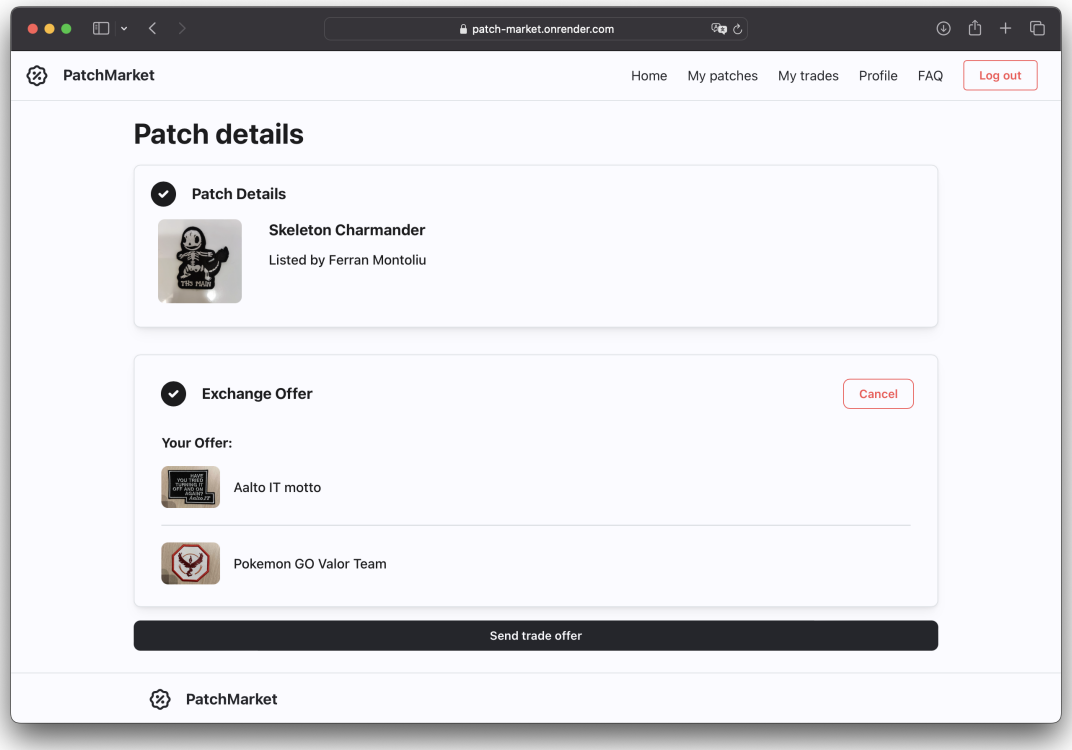


Figure 23: Offering a Trade - Desktop

5.1.6 Patch details

The patch details page (mobile: Figure 18, desktop: Figure 21) allows users to visualize the picture, name, description, university of origin and categories associated to a patch. Additionally, the *Share* button enables users to copy the link of the patch and share it with friends. Furthermore, users can also see the name and surname of the trader who listed the patch and start a trading offer by clicking on the *Make an offer* button.

5.1.7 Patch details - trading

By clicking on the button: *Make an offer* in the patch details page, traders will see the screen illustrated in Figure 19 and Figure 22. This page is still part of the patch details page and presents a collapsed version of the patch details which only shows the picture, name and owner of the patch. This allows the user to visualize them at any time while choosing what to offer in exchange. In this page users will see a list of their patches that they can browse to quickly find the patches they would like to trade in exchange for the chosen patch. The patches they select from the list will be listed in an apposite area at the top of it using chips. In this way, they can easily be visualized and also removed by clicking on the x button that is available in each chip, in case users decide to remove them. This functionality prevents them from having to search for the patches again to deselect them or remember which ones they had chosen. The *Confirm* button at the top allows to confirm the offer and navigate to the screen shown in Figure 20 and Figure 23 which gives users a recap of their trading offer and allows them to cancel it with the *Cancel* button in case they would like to select different patches, or to send it to the other trader by clicking the *Send trade offer* button.

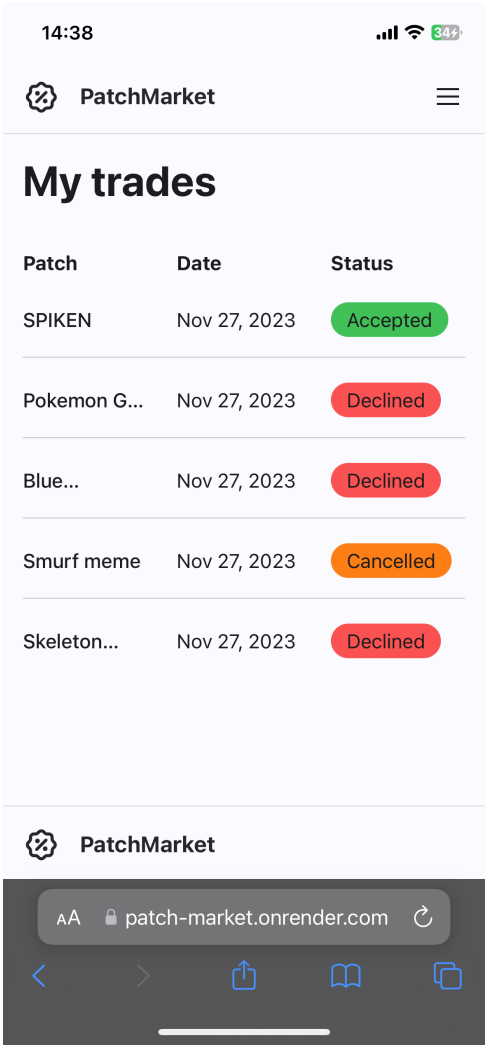


Figure 24: Trade History

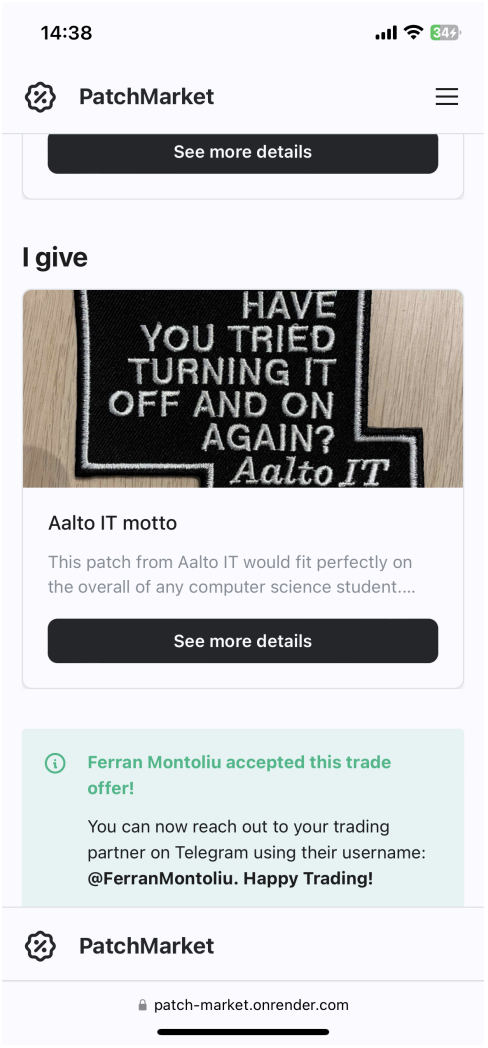


Figure 25: Trade Details

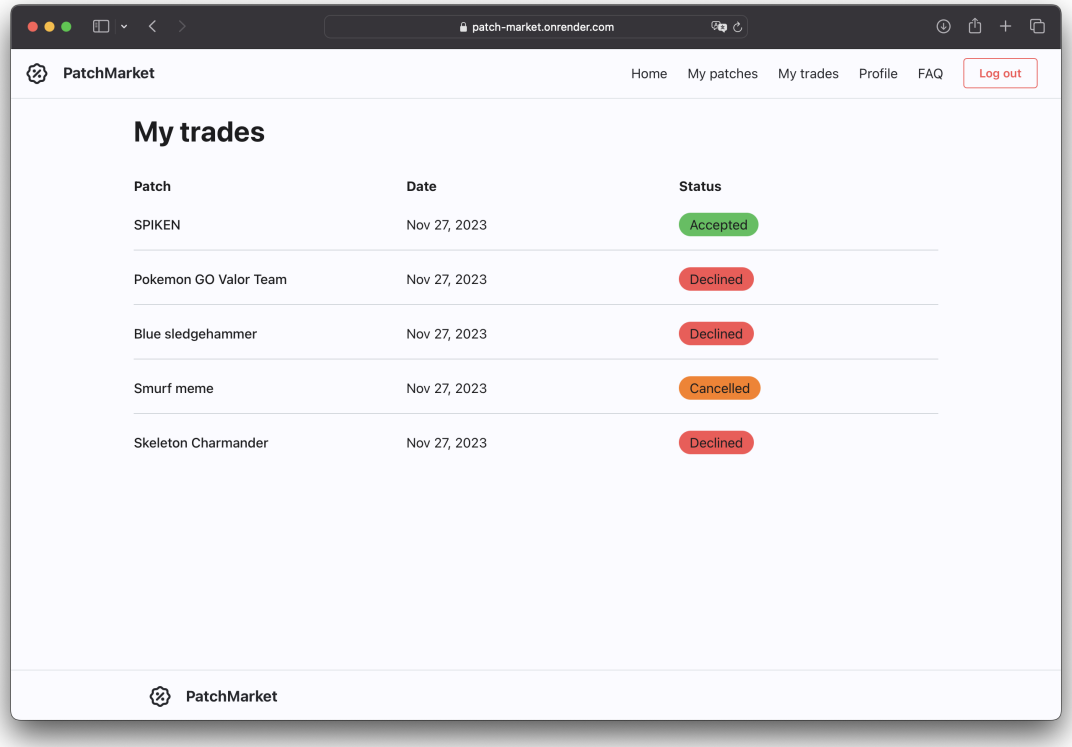


Figure 26: Trade History - Desktop

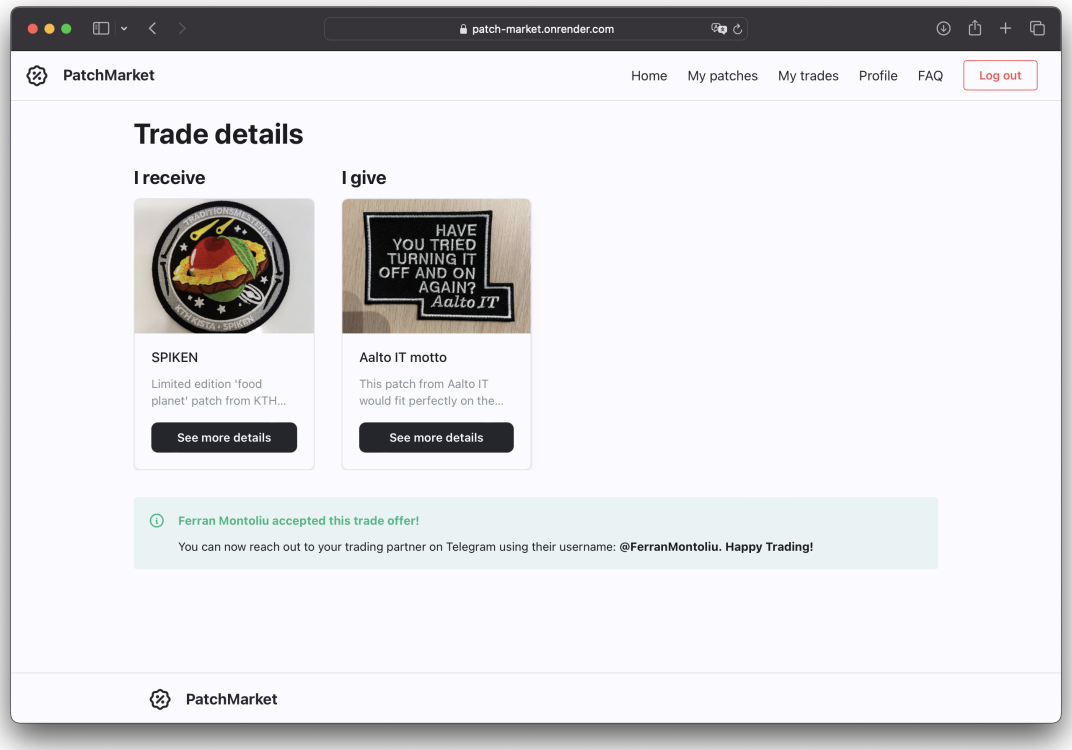


Figure 27: Trade Details - Desktop

5.1.8 Trade history

The trade history page illustrated in Figure 24 and in Figure 26 allows traders to visualize a list of their past and ongoing received and sent offers. The list includes the name of the patch and the date of the offer, but also shows to the user the state of such offers using suitable colors to allow them to quickly visualize it. By clicking on any of the offers, users will be able to see its details and take an action in case the offer is still pending.

5.1.9 Trade details

The trade details page (mobile: Figure 25, desktop: Figure 27) allows users to visualize the patches they would give away and receive if a specific trade goes through. If the offer was received by another trader, they can choose whether to accept it or decline it. If the offer was sent by them, they can instead choose to cancel it or keep on waiting for an answer.

All these operations can be performed by using the buttons at the bottom of the page. The *Accept* button is indicated in green, while the *Cancel* and *Decline* buttons are red. If users decide to accept a trade offer, all the other offers for that patch will automatically be declined, and all the offers that the user made to others which include that patch will be automatically cancelled. This mechanism ensures that once the patch does not belong to a user anymore, they cannot continue to trade it. Before taking any action, users are informed about its effects through alerts and can thus take an informed decision.

The trade details can also be visualized for trade offers that have already been accepted, declined or cancelled. In such case, users will only see a recap of the offer and information about its status, but no buttons to interact with it. Additionally, if the offer was accepted, traders will also see each other's Telegram accounts that they can use to contact each other and decide on the practicalities of the trade.

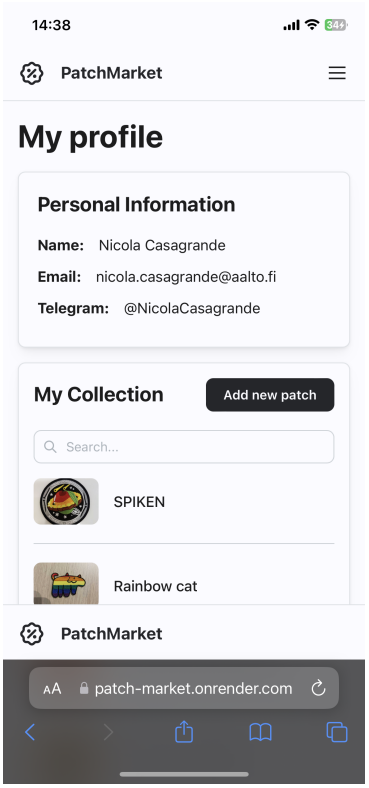


Figure 28: Profile

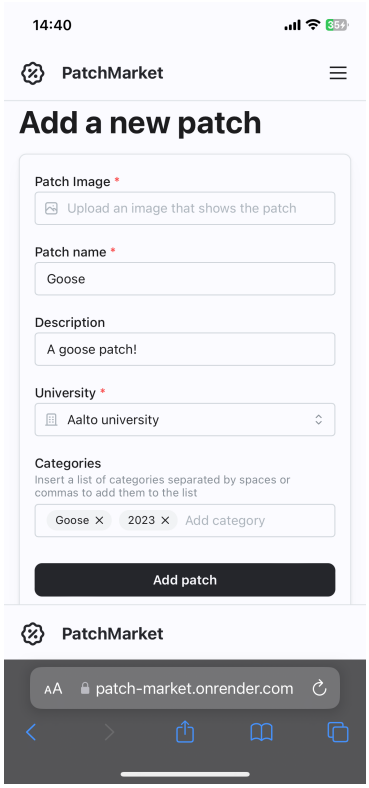


Figure 29: Add a patch

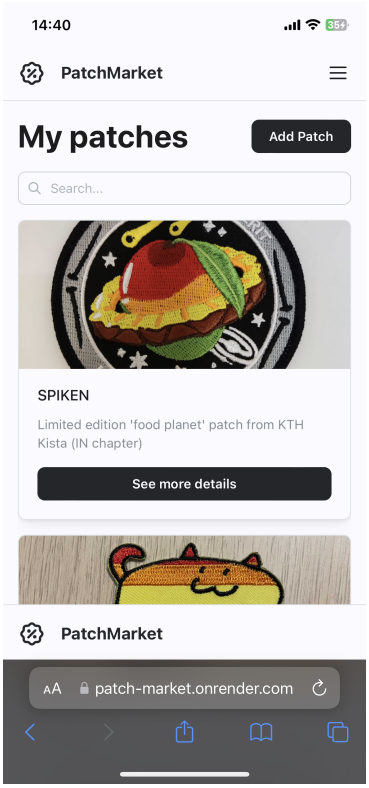


Figure 30: My Patches

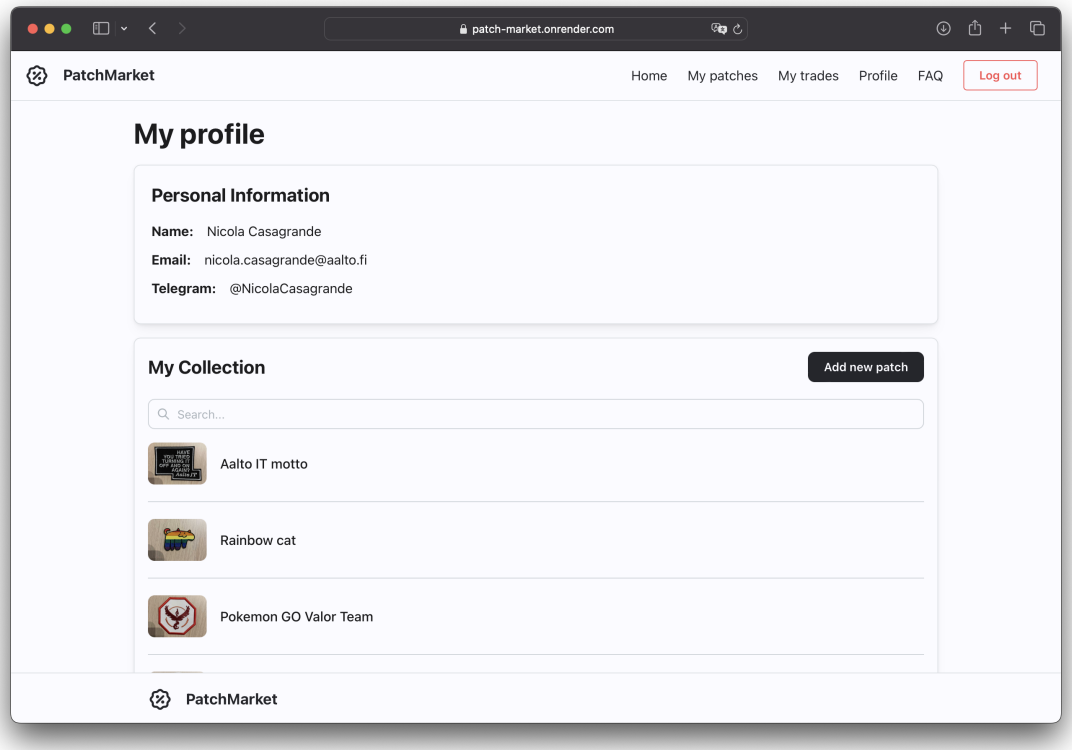


Figure 31: Profile - Desktop

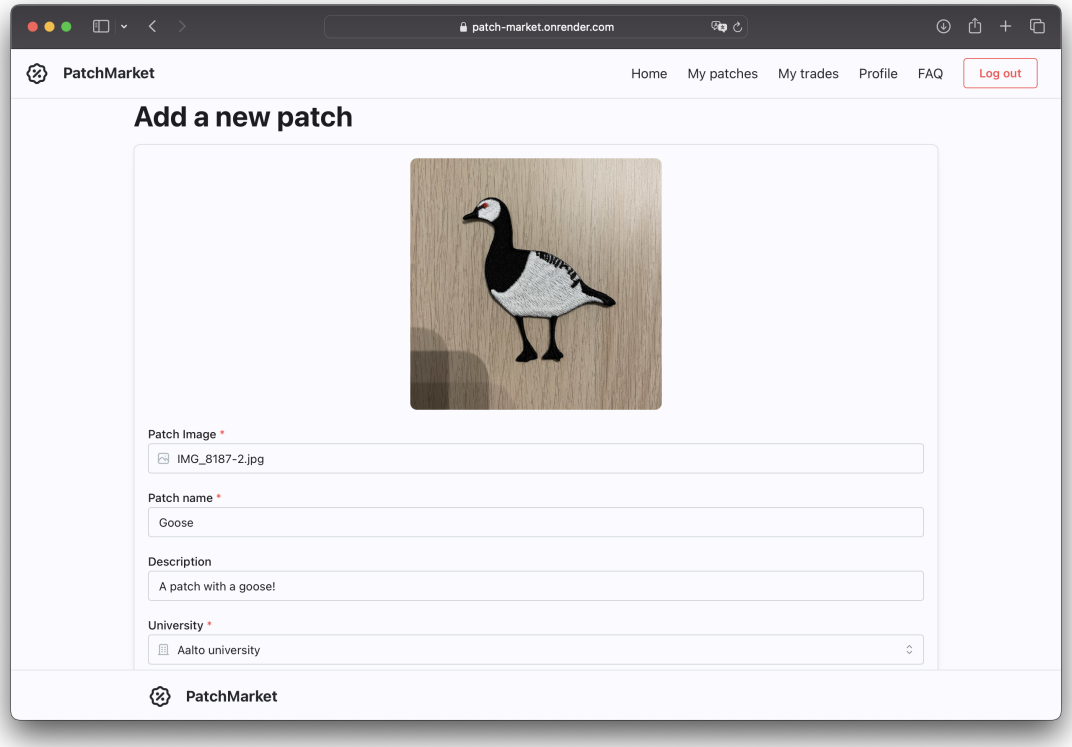


Figure 32: Add a patch - Desktop

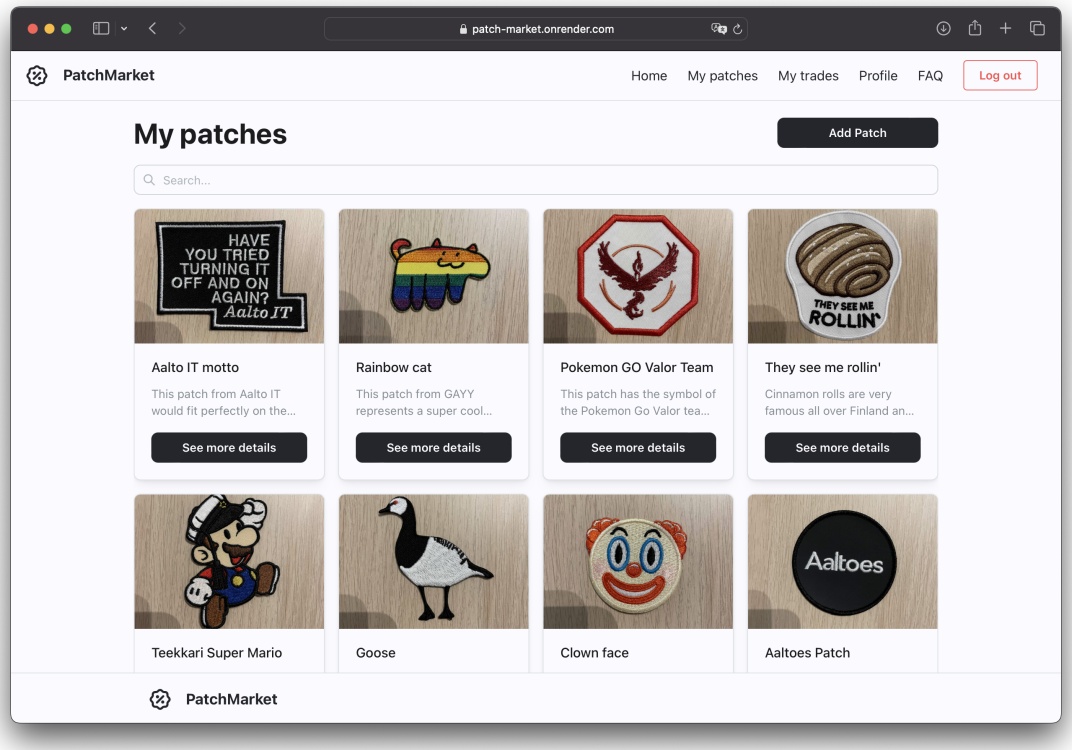


Figure 33: My patches - Desktop

5.1.10 Profile

The profile page (mobile: Figure 28, desktop: Figure 31) allows users to visualize their profile information and their collection of patches. The button *Add new patch* allows them to navigate to the *Add a patch* page and add a new patch to their collection. The *My collection* section can also be visualized in the *My patches* screen which is illustrated in Figure 30 and Figure 33. This screen is almost identical to the home page (mobile: Figure 13, desktop: Figure 16), but it enables users to browse the patches they own, search through them by name and add new ones by clicking on the *Add patch* button.

5.1.11 Add a patch

The page that allows users to add new patches to their collections is illustrated in Figure 29 and Figure 32. It contains a file picker that can be clicked to select a picture from the device gallery and add it as a picture for the patch. Once the image is added, it is shown in the page above the file input field. The *Add a patch* screen also contains text fields for the patch name and description, and a field to enable users to select the university of origin for the patch from a list that is retrieved from the database. Categories can easily be added to the patch by writing their name in the apposite field and pressing the space or enter button on the keyboard. Finally, the *Add patch* button at the bottom allows to confirm the operation and add the patch to the collection. Errors in the form will be indicated to users with a red textual description under the field that contains the error, so they can easily fix it.

5.2 Fonts and colours

The fonts and colors that are utilized in the PatchMarket application were chosen to provide to the users the best possible user experience from both mobile and desktop. We chose to use the font **Roboto** which is designed by Google to provide great readability and performance in web applications. Roboto is also a modern and minimal font that gives a clean look to our application. Finally, Roboto provides a consistent bold variant, which we used to highlight important information in the screens. To keep the application consistent and aesthetically pleasing, we only used a single font throughout all the screens.

Regarding the colors, PatchMarket is designed following a very simple color palette. The application mainly makes use of the basic colors: black and white, which make it look clean and modern. The high contrast between the two colors makes the information easy to read on the screen. Other colors we used are red and green, these colors enable us to indicate to the users the options they have with more clarity. We followed standard guidelines and used red to highlight the cancel and decline buttons and green to highlight the accept button. Finally, we also make use of a different set of colors to differentiate categories, menu options and the status of trade offers, so users can easily distinguish between them.

6 Technologies and architecture

6.1 Dynamic functionalities

The dynamic functionalities of the service are divided into two categories.

Functionalities that are available to non-logged-in users:

1. **Sign up (register):** a user inputs their data and, after validating that the data is complete, a new user is inserted in the database using said data.

2. **Log in:** after a user has created an account, they can log into the platform. To do so, they need to input a valid e-mail and password. This information is used to check if a user with those credentials exist in the database. If so, the user is logged in and a token is generated for that session, in order to securely authenticate the user. If the information is not correct, the user is not logged in.

Functionalities that are available to logged-in users:

1. **Log out:** after having logged into the application, a user can log out. This will invalidate their token.

2. **See profile information:** a user can see the information they used for registration.

3. **Search for patches:** see all the patches that are listed for trading by other users, as well as search for specific patches by name or category. This is done by querying the database with different query parameters.

4. **See patch details:** see all the details about one patch. The information is fetched from the database and displayed nicely.

5. **Upload a patch:** a user can upload as many patches as they want. After this data is validated, the patch is uploaded to the database. For the categories, if the user has introduced categories that already existed, they are assigned to the patch, but if the user has used keywords that were not existing categories, they are created in the database and assigned to said patch.

6. **List a patch for trading:** after creating a patch, a user can choose if they want to list that patch for trading or not. If the patch is listed for trading, it will be displayed in the marketplace (home screen), else it will be hidden.

7. **Accept or decline received exchange offers:** when an exchange offer is sent to a user, this user can choose to accept or decline this offer. This will update the status of the trade offer in the database and, if the transaction is accepted, it will update the ownership of the patches in question, as well as remove the patches from the marketplace (mark them as not available for trading). If the offer is declined, the ownership remains the same and the patches are still up for trading.

8. **See history of previous and ongoing trade offers:** a user can see all the previous and ongoing trade offers, including accepted, declined, pending and cancelled offers. This is done by querying all the trade offers from the database where the user was either the source or the recipient.

9. **See exchange offer status:** a user can see the status of all the past and ongoing trade offers in which they have been involved.

10. **Send exchange offers:** a user can take a patch from the marketplace (listed for trading) and send an exchange offer to trade it. This is done by creating a new trade offer in the database.

11. **Cancel sent exchange offers:** a user can cancel any outgoing exchange offer made at any time, while this trade offer is still pending. This is done by checking if the status of the trade is still pending, and, if it is, updating the status to cancelled. This status is final, so the user that received the offer will not be able to accept or decline it anymore, since it is already cancelled.

6.2 Technology choice

For the technology stack, we decided to go for some of the most popular languages, libraries, and frameworks right now in the market, as they are not only good for this use case, but also very widely used in the industry.

For this system, we'll have a web-based single page application, which will consist of two components: a front-end and a back-end (an API).

For the **front-end**, we decided to use the following libraries and frameworks:

1. **React** for building the UI.
2. **TypeScript** to make the application more robust and resilient. Even though TypeScript may slow us down during the development, as we'll have many more compilation errors, if we are strict about the types, we'll have essentially no runtime errors, thus making our application much more robust.
3. **Mantine** for pre-made responsive components in the UI, and also for its form management system.
4. **React router** for the navigation between pages.
5. **React query** to fetch the data from the API and handle the different states we encounter while working with an asynchronous API (loading state, error, no data, etc.).
6. **Eslint** to ensure code quality, readability, and maintainability.

For the **back-end**, we decided to use the following libraries and frameworks:

1. **Node.js** as the JavaScript environment. We wanted to use Bun, as it's faster, more secure, and it provides things that Node.js does not provide. Nevertheless, after some analysis, we saw that it was not production-ready yet (even after the v1.0.0 release), as it's still lacking many methods from the Node.js API.
2. **Express** to create the infrastructure needed to implement the required API endpoints and middlewares.
3. **TypeScript** to make the application more robust and resilient. Even though TypeScript may slow us down during the development, as we'll have many more compilation errors, if we are strict about the types, we'll have essentially no runtime errors, thus making our application much more robust.
4. **MongoDB** as our database. Here is where we will store all the information and the images. We'll not be using a file system for the files, since we will only be storing images, and we'll restrict the size of those.

5. **Mongoose** to query the database, cast the values, and validate them. It's basically an ODM to interact with the database with less boilerplate.

6. **JSON Web Token** for user authentication. This allows us to authenticate users through a token, and close their session after some time, thus providing an extra layer of security to our application.

7. **Bcrypt** for encrypting the passwords from the users in a secure way.

8. **Eslint** to ensure code quality, readability, and maintainability.

6.3 Architecture and structure of the database

We use MongoDB as our database system, which is a NoSQL document-based database. Nevertheless, MongoDB provides the possibility of handling relationships between entities.

That being said, we'll be using the following architecture for our database implementation:

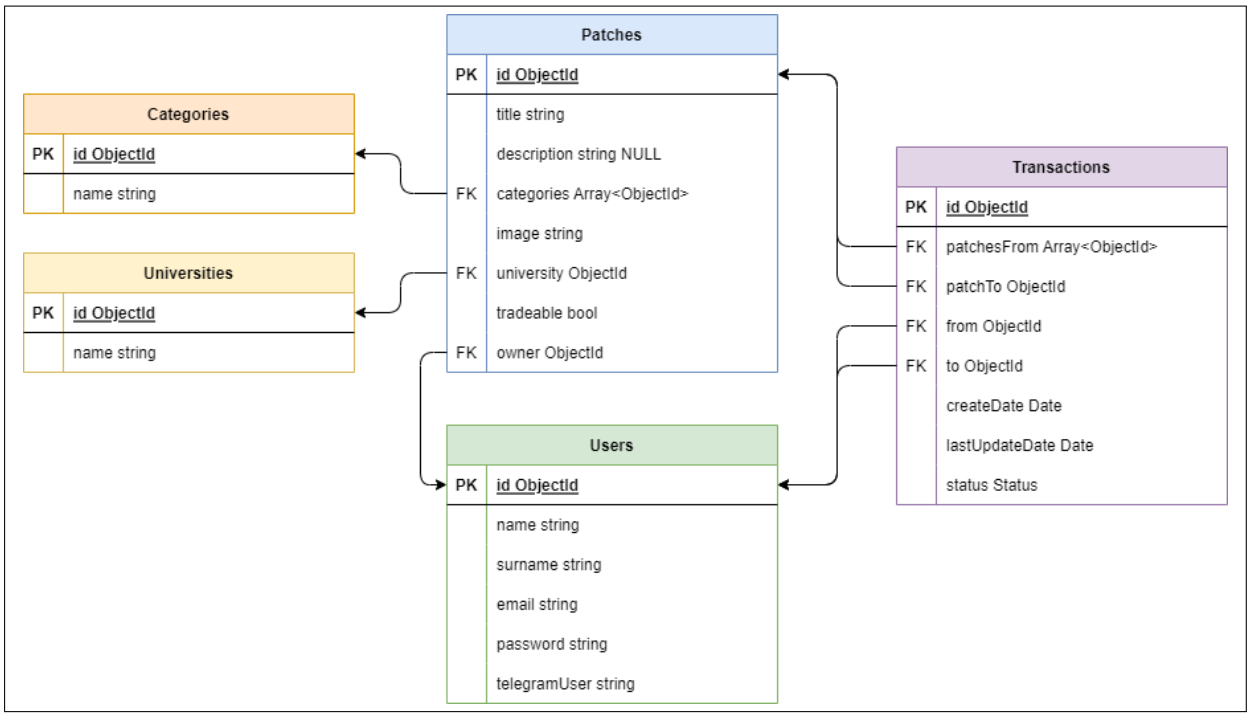


Figure 34: Database entity relationship diagram

As it can be seen, the entity relationship diagram is very similar to one from a SQL database. We followed this approach in order to avoid having repeated data, and because this makes it easier to migrate to a SQL database in the future if needed.

In this case, we have 5 collections, with the following properties:

1. **Patches:** stores information about patches.
   (a) **id:** identifies the document. Therefore, it is a unique field.
   (b) **title:** title or name of the patch.

(c) **description:** description of the patch. This field is optional.

(d) **categories:** array of categories for the patch. There can be 0 to N categories.

(e) **image:** image of the file. The image is stored directly in the database in Base64 format.

(f) **tradeable:** this determines if a patch is up for trading or not.

(g) **university:** the information about the university of origin of the patch.

(h) **owner:** the information about the owner of the patch.

2. **Users:** stores information about users.

   (a) **id:** identifies the document. Therefore, it is a unique field.

   (b) **name:** name of the user.

   (c) **surname:** surname of the user.

   (d) **email:** e-mail of the user. This is used to authenticate the user, and therefore is unique.

   (e) **password:** hashed password of the user. This is used to authenticate the user.

   (f) **telegramUser:** telegram username of the user.

3. **Transactions:** stores information about patch trades between users.

   (a) **id:** identifies the document. Therefore, it is a unique field.

   (b) **patchesFrom:** the patches offered in exchange for a specific patch.

   (c) **patchTo:** the patch that is listed as tradeable and to which a user has made a trade offer.

   (d) **from:** the user that makes the offer.

   (e) **to:** the user that receives the offer.

   (f) **createDate:** the date in which the trade offer was made.

   (g) **lastUpdateDate:** the date in which the trade offer was last update (updated its status).

   (h) **status:** the status of the transaction. It can have one of the following values:

      - **Pending:** the offer has been made, but no response has been given yet.
      - **Accepted:** the offer has been made, and the user that received it has accepted it.
      - **Declined**: the offer has been made, but the user that received it has rejected it.
      - **Cancelled:** the offer has been made, but the user that made it has cancelled it.

4. **Categories:** stores information about categories. A patch can have one or more category.

   (a) **id:** identifies the document. Therefore, it is a unique field.

(b) **name:** name of the category. This name is unique.

5. **Universities:** stores information about universities. A patch can have one university of origin.

   (a) **id:** identifies the document. Therefore, it is a unique field.

   (b) **name:** name of the university. This name is unique.

6.4 Security

In order to improve the security of our service, we provide the following layers of security:

1. **Encryption:** we do not store passwords as plain text. Instead, we encrypt them using a hash function. The only time in which a password is sent as plain text is when registering a new user, but even then the password is sent through HTTPS, so the content of the HTTPS packet is encrypted. For hashing the password we use bcrypt, which is a library that provides secure hashing methods, which are not susceptible to timing attacks, and provides 60-character hashes.

2. **SSL/TLS:** thanks to an SSL certificate, we can make sure we make our platform accessible through HTTPS, thus securing the connection to our system.

3. **JWT**: Json Web Tokens are used to authenticate our users. The way this works is that, when a user logs in, a token is generated for said user. All the request this user makes have to contain that token as a header in order for us to authenticate them. If a call has no token, then it is not processed. Also, if a call has an expired token (the user has been inactive for too long), we will also not process it. In this case, the user will have to log in again in order to refresh their token. This is a system widely used in many web applications nowadays, since it protects both the user and the API from malicious attacks.

4. **Up-to-date libraries:** we use the latest stable version available of all the libraries we import in the code. By doing so, we make sure that we have the latest security patches.

5. **Custom error pages:** through the default system error pages (for example 404 error page), attackers can fetch information about the system that is running our service. By creating custom error pages, we can hide most of that information, so it's not available to the public, thus making it more complicated for attackers to get information about our system.

6. **Double validation in form:** the data sent through forms is validated in both the front-end and the back-end to ensure its contents are what they are supposed to be. Also, all the contents inside the forms are escaped, so SQL injections or other similar attacks are impossible to execute in our service.

7 Project design

7.1 Breakdown structure for the project

To complete our project, we decided to break down the workload that is needed for the implementation into small tasks and rely on the **Scrumban** project management methodology. Scrumban is a hybrid system between Scrum and Kanban that allows to approach the implementation of a project in a way that gives both the predictability of Scrum and the flexibility of Kanban.

Unlike Scrum, Scrumban does not use fixed-length sprints, but it relies on a Kanban board to organize the tasks. This methodology allowed us as a team to start new tasks whenever we had the capacity to do so instead of having to follow fixed sprint deadlines. In particular, we decided to utilize a Kanban board with three stages which are: *To Do*, *In progress* and *Done*. When a member of the team could start a new task, they took the first task from the *To Do* column and moved it to the *In progress* column to indicate to the other members that they were working on that specific task. When the task was completed, it was moved to the *Done* column.

The *To Do* column acted as a common backlog that was shared between the team members. The tasks in the backlog were ordered by their priority so that the first task was always the one with the highest priority and that should be started first. To decide on which new tasks to put in the common backlog, the team organized weekly meetings where we decided on which new tasks had to be added to the backlog and their time estimates. This approach demonstrated to work well, as we managed to complete all the features that we planned for the project on time.

7.2 Task estimation and prioritization

In Table 1, we illustrate all the tasks that we completed for the implementation of Patch-Market. Each of the tasks comes with a priority and the number of hours that the task took us to complete. The priority of a task is determined by a number between 1 and 5, with 1 indicating the highest priority, and 5 the lowest one. Even though Table 1 is not indicated in red, the table has changed from our planned tasks schedule, as we added new tasks when we realized the need for them during the development phase.

Page	**Task name**	**Priority**	**Time**
-	Prepare project structure	1	6h
-	Prepare front-end for react query	1	2h
-	Error 404 page	4	30min
-	Prepare back-end skeleton and database setup	1	8h
-	Add notifications to all actions	3	2h
-	Fix overall code style	3	2h
-	Deploy	1	2h

Page	Task name	Priority	Time
Login	Implement UI	1	1h
	Check validity of user data	2	1h
	Implement front-end login functionality	1	1h
	Implement back-end login functionality	1	2h
Sign Up	Implement UI	1	2h
	Check validity of user data	2	1h
	Implement front-end and back-end sign up functionality	1	4h
Logout	Implement front-end logout functionality	5	1h
	Implement back-end logout functionality	5	1h
	Implement automatic logout for users with an expired token	5	2h
Home	Implement UI	1	4h
	Implement search	4	2h
	Fetch and display tradeable patches	1	2h
Menu	Implement UI	2	2h
	Make menu close automatically	5	30min
	Implement Navigation	1	2h
Patch details	Implement UI	1	4h
	Retrieve patch details and display them	1	4h
	Implement share a patch	5	30min
	Implement API and UI to make a patch tradeable	1	2h
	Implement API and UI to unlist a patch	4	1h

Page	Task name	Priority	Time
Patch details (Prepare exchange offer)	Implement UI	1	4h
	Retrieve list of owned patches	1	1h
	Implement chips for selected patches	4	2h
Patch Details (Send trade offer)	Implement UI	1	2h
	Add exchange offer front-end and back-end	1	4h
My trades	Implement UI	2	4h
	Retrieve list of trade offers from the database and display them	2	2h
Trade details	Implement UI	1	2h
	Retrieve offer details from the database and display them	1	2h
	Accept exchange offer back-end	1	2h
	Add a way to see user information after a completed trade	1	2h
	Unlist traded patches	2	1h
	Cancel or reject all transactions including a patch when that patch is traded	2	4h
	Implement change offer status front-end and back-end	1	4h
Profile	Implement UI	4	1h
	Fill profile with user data	4	2h
	Retrieve user patch collection	4	2h
	Implement search functionality	5	2h
Add a patch	Implement UI	1	4h
	Implement add patch front-end and back-end functionality	1	8h
My patches	Implement UI	5	2h
	Fetch patches from the database and display them	5	2h
FAQ	Implement UI	5	2h

Table 1: Description of all tasks, grouped by page.

8 Project Timeline Overview

The project schedule was initially divided into three different phases. The design phase marked the beginning as we conceptualized the platform, outlined its features, and defined the scope of the project. In the following phase, the design decisions were presented, and a first prototype was shown and critically evaluated at the gala. This provided us with valuable feedback, which we adapted in the third phase.
The implementation phase consisted of implementing the web service and documenting the final report. A small buffer was placed at the end of the schedule to allow for potential debugging or code fixing. A comprehensive Gantt chart illustrating the planned schedule can be found in Figure 35.
As shown in the Gantt Chart (Figure 35), all planned project milestones have been successfully achieved. The Demo Gala played an important role in the planning, providing insightful feedback that was carefully considered in the subsequent phases and also changed our planning. In our efforts to improve the project, we took into account some of the feedback in our application, while others exceeded the project's scope or were unfeasible within the given time. Further details regarding the feedback and our responses can be found in the reflections from the demo phase section (subsection 9.1) and are also elaborated upon in the Future Work (subsection 9.2) section of this report.
For a better comprehension of the project's progress, we opted to separate the Frontend and Backend components during the implementation phase. This strategic decision allowed us to allocate additional time to properly address the Backend implementation.

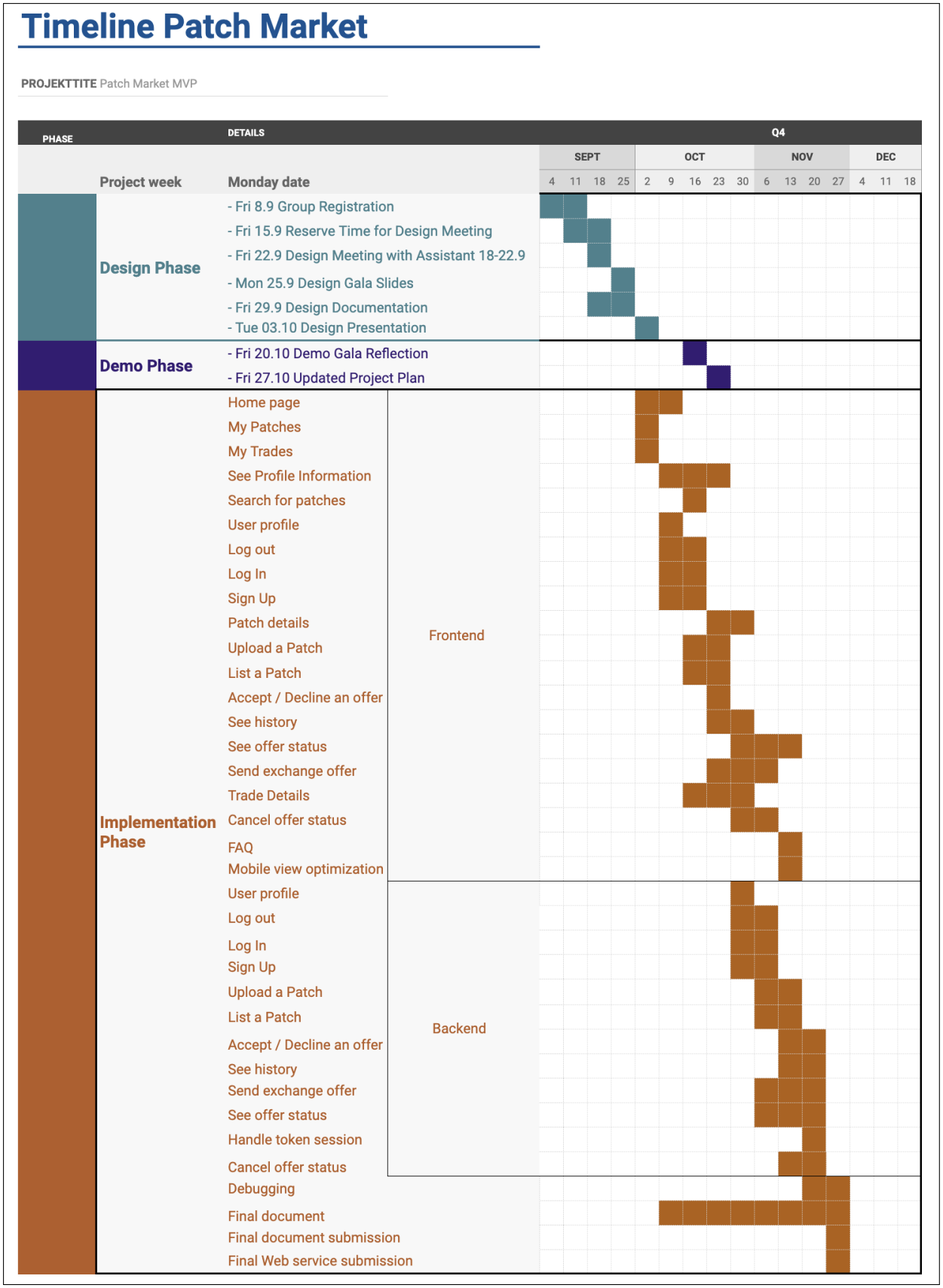


Figure 35: Project timeline overview - Gantt Chart

8.1 Responsibilities for each group member

In the PatchMarket team, every member was equally responsible for the project development. We decided that, to ensure the best cooperation between us and contribute to the work of each other with our own ideas and skills, all the responsibilities were shared between the team members. Since we used the Scrumban approach, we did not assign specific tasks to each of the members beforehand. Instead, each of us started to work on the task with the highest priority that was available in the shared backlog. Despite the shared responsibility, we designated one team member to be the final responsible for the three key areas of the project implementation. In particular, Lars Oberhofer was responsible for the user experience of the application, the user research and the documentation. Nicola Casagrande was responsible for the implementation of the front-end of the application and Ferran Montoliu Boneu was responsible for the implementation of the back-end of the application. During the Design and Demo phase, although one team member was designated as responsible, the entire team actively collaborated on all aspects of the implementation. Following the Design and Demo phase, we refined our responsibilities to better focus on the implementation phase. Lars Oberhofer became responsible for the back-end development, and Ferran Montoliu Boneu took on a full-stack role, containing both front-end and back-end development. This adjustment ensured the robust and punctual completion of the implementation phase in our application.

9 Final reflection

In this section, we present the final reflections, preceded by the reflections from the demo phase, and the future work that we have planned for the project.

9.1 Reflections from the demo phase

During the demo gala, we presented an early stage of the PatchMarket application, and we received valuable feedback from other groups. The feedback was especially useful because the people providing it are students and part of the user group for our application. The Demo gala prompted us to reflect on features we had not initially planned to implement in our application but found useful for the end users. We also reconsidered some ideas we had contemplated but excluded due to time constraints during the development phase. The most significant insights from the demo are as follows:

1. **Rating users:** to enhance the trading experience and build trust among users, we considered allowing users to rate each other on the platform. This feature could help users gauge each other's trustworthiness by providing feedback on their trading experiences. Although we initially thought of this during the project's ideation stage, we deferred its implementation in the first version due to time constraints.

2. **Categories:** we received a suggestion to let users create personalized categories for patches, as predefined categories may not cover all potential categories, and providing users with the flexibility to create their own categories could greatly enhance their experience. This feature aligns with our initial design for PatchMarket, where users can choose between a set of already defined categories (created by other users), or create their own (which will be made available to other users), so it is included in the first version of the project.

3. **School of origin for patches:** adding information about the school of origin for patches was suggested to provide students with crucial details about the patches they intend to trade. This feature is relatively quick to implement and promises to significantly improve the user experience, so we decided to update the original plane to include this for the first version.

4. **Sharing patches with friends:** although users can currently share patches by copying and pasting the URL of each patch, the feedback indicated that this may not be clear enough. Because of that, we decided to add a dedicated button for sharing patches via social media to improve usability in the patch details page.

5. **Colorful UI:** the PatchMarket UI, designed for simplicity and user-friendliness, currently maintains a minimalistic, serious, and trustworthy appearance. While we received feedback suggesting a more colorful UI, we decided that this change does not align with our vision for the service, so we decided to keep the current color palette.

6. **Other questions:** various questions arose in the feedback session, such as how we ensure users complete trades once accepted and how users can communicate for trade agreements.

   Given the course's scope, we decided not to address these issues directly within the application, as they would expand the project's scope significantly. However, we decided to include the contact information (name and Telegram username) of the other user when a trade offer is accepted. This enables users to message each other through Telegram and arrange trade terms independently.

9.2 Future work

In this section, we present the future steps that we want to take for this project. The functionalities listed below are not a part of the final submission, since we considered they would be out of scope, but we wanted to document them as they are part of our long-term road-map:

1. **Rating and reporting users:** in the MVP, users have to blindly trust other users, since there's no way to rate or report a user. This means that, if a user has not fulfilled a trade they agreed to fulfil, the other user has no way of rating the other user. This is a huge issue because this could mean that we might lose the trust of our users, since we don't have any system to punish bad behavior from users.

2. **Chatting with users:** in the MVP, in order to contact a user, one has to send them a Telegram message, but it would be nice to have a chat inside the platform, so our users wouldn't be forced to use another application in order to finish a trade.

3. **Making sure trades between users actually happen:** in the MVP, we trust that when 2 users agree on a trade, they will actually finish that trade in real life, but we don't have any system in place to verify that. It would be cool to have a system to verify that. This would also make the rating and reporting system much more reliable, since we would have more insights on if the trade happened or not.

4. **GDPR compliance:** in the MVP, we are not taking into consideration GRPR regulations, since we don't have a privacy policy, cookies policy, or legal disclaimer. In order to comply with the current European regulation regarding data privacy, we should have all this information easily accessible in the webpage.

5. **Internationalization:** in the MVP, the platform can only be found in English, but we could easily translate it to other languages, like Finnish, German, Spanish, and Italian, amongst others.

6. **Banning inappropriate language and images:** in the MVP, one can upload any picture they want with any patch they want. We trust our users to make good use of our platform, but we should be prepared for the worst in future versions. That's why we should establish a system to ban inappropriate language and images from the platform (patch name, patch description, category, chat, etc.).

7. **Optimization of the deployment:** when the service is in a more advanced stage, we are planning on deploying it in our own VPS (Virtual Private Server) so that we can have total control over the underlying technologies and resources dedicated to it, as well as the accepted security protocols and headers, along with other crucial configurations.

8. **Automated tests:** due to the limited scope of this project, we decided not to implement any automated test or any CD/CI pipeline. Nevertheless, in order to improve the maintainability of the platform, we have to add automated unit, integration, and end-to-end tests.

9.3 Final reflections

After completing the PatchMarket MVP, we are proud to say that we were able to do all the things that we initially planned to do, and even some more that we decided to do later on, which are explained in subsection 9.1.

Even tough there is still work to do to consider this a production-ready service, as can be seen in subsection 9.2, we are happy with the current implementation, as it runs smoothly and without any runtime error that we are aware of.

In the process of developing our project, we encountered significant challenges in mastering the technology stack for both the front-end and the back-end. The languages, frameworks, libraries, and architectures used initially presented a steep learning curve for our team. However, we tackled these challenges head-on by implementing a well-structured organization and training strategy. Our team members dedicated themselves to learning the new technologies and, as a result, we not only overcame the initial difficulties but also achieved proficiency in the technology stack. This dedication and adaptability have culminated in the development of a robust and efficient code base that stands as a testament to our team's hard work and collaborative spirit.

In conclusion, the successful development of the PatchMarket MVP exemplifies our team's adaptability in overcoming the steep learning curve of the technology stack, since we have exceeded our initial goals, resulting in a smooth, error-free implementation.

10 Focus area

Our focus area is the UI and UX of the web service. To accomplish this, we have divided it into the following challenges:

1. **Responsive and mobile-first:** we created a platform which is intended to be used both with a smartphone and a desktop device, but mostly in the first one. For this, we have focused our design and implementation on making it mobile-first, while keeping it fully responsive, in order to have a visually appealing user interface and a smooth user experience.

2. **Professionalism:** we want our product to look as professional as possible. For this reason, we followed already established industry guidelines and practices both when designing and implementing our product so that it looks as professional as possible.

3. **Micro-interactions:** to ensure the best user experience, we communicate everything with the user through micro-interactions. Examples of this are:

   (a) When a user completes an action, a notification will pop up.

   (b) When the page is loading, the user will see a loader.

   (c) When there is no data, the user sees a message indicating that there is no data instead of a white screen.

   (d) When there has been an error, the user will be shown a specific message indicating what happened.

   (e) The forms have validations both in front-end and in back-end and, when the form contains an error, the error is specific and is placed below the field that has the error. Also, if a form is missing any required field, the user cannot submit the form.

References

The carolina trader. (n.d.). Retrieved from https://www.thecarolinatrader.com/products

Copyright directive. (2000). Retrieved from https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32000L0031&from=EN

ebay. (n.d.). Retrieved from https://www.ebay.com

Etsy. (n.d.). Retrieved from https://www.Etsy.com

Patch-collection. (n.d.). Retrieved from https://www.patch-collection.com/trade-list/

Reddit trade patches thread. (n.d.). Retrieved from https://www.reddit.com/r/patchtrading

Scout jamboree events. (n.d.). Retrieved from https://www.scoutshop.org/blog/patch-trading-at-the-2023-national-scout-jamboree.html