ECSE 456 - Turbodega - DP 03 Final Report

Project Supervisor: Professor Daniel Varro (daniel.varro@mcgill.ca)

Advisors: Julio Castanedaescobar (julio.castanedaescobar@mail.mcgill.ca)
Daniel Franco (daniel.franco@mail.mcgill.ca)

Yahya Azami
yahya.azami@mail.mcgill.ca
260535376

Aliah Mohd Nazarudin
nur.mohdnazarudin@mail.mcgill.ca
260658075

Nabil Ersyad Noor Eddie Putera
nabil.nooreddieputera@mail.mcgill.ca
260675196

Thusha Sivapatharajah
thukarasha.sivapatharajah@mail.mcgill.ca
260687292

Abstract

Turbodega's web application is intended to help bridge the gap between small grocery owners in emerging countries and big supermarkets in terms of logistics and purchasing power. This is done by allowing the former to tap into modern retail logistics and economies of scale by providing a platform for consolidating their orders and allowing bigger distributors to bid on these. This report accounts for the progress made on building the web application this semester, detailing the technology stack and the software architecture we're using, how the project is managed and what are our plans for the upcoming semester.

Contents

1 Introduction

In Latin America, there are more than two million small grocery stores located in marginalized and poor zones, and both owners and customers belong to a low-income population [1]. These small businesses provide daily sustenance for about ten million people in this region. The grocery stores are usually owned by the head of household and most of them are owned and managed by women [2]. In the past decades, these grocery stores have experienced losses in sales and in profit margins. This is due to the growth of supermarket, auto service shops and convenience stores, including changes in consumption patterns of the populations [1]. Three factors that led to this issue are competitive prices from the bigger stores, the smaller network of accessible distributors and inefficient use of retail-oriented technology [3].

Competitive prices from bigger stores happen because grocery stores usually order the goods from the distributors in smaller quantities. Ordering in small quantities force them to pay a higher price to the distributors. In addition, they also have a limited pool of distributors, so they do not have another choice and agree to the price set by the distributors. From the distributors' point of view, they also have a limited pool of small grocery stores that they can reach out to. Hence, for their benefit, they supply the goods at higher prices since the grocery stores are ordering from them in smaller quantities.

Another issue raised in small businesses is the inefficient use of retail-oriented technology. Owners do not manage their stores properly because they often do not update their inventory list regularly and they only order items when they are almost out of stock. On the other hand, the distributors are struggling to keep track of small orders from multiple stores at the same time.

Considering these issues, Turbodega is taking the initiative to be the middle ground between these two entities. Our aim is to create an accessible platform for grocery store owners and the distributors to create a level playing field in the market for grocery goods. As a result, this will increase profit margins and efficiency of small grocery stores.

2 Background

Turbodega is a start-up founded by two McGill MBA students from Mexico and Peru. Its aim is to develop software geared technology toward users in the small grocery store industry in Latin America. Our team is one of the two teams working with the founders to help them bring their vision to life. Specifically, we have been tasked with developing a website for them. Since most of us are in electrical engineering, it was important to familiarize ourselves with the workflow of a software development project and the necessary technologies to be used.

In addition, it was also important to understand the background of the target users and the requirements for the website. The team had to understand the business flow between the grocery stores and their suppliers. Then, we had to know the basics of what is expected from an auctioning website to build the required pages.

Other things that need to be identified are the languages to be used, code organization, fundamentals of web development and tasks related to the front-end and back-end. GitHub was used since every member had some level of experience using it. However, it was necessary to gain the habit of focusing development on a GitHub repository and storing every important component of the project on it. This was also made more important after the team decided to use the GitHub domain servers to host the website where it can be accessed on the internet.

This phase of the project is more focused on front-end development. Therefore, members were all tasked to familiarize themselves with HTML, CSS and JavaScript languages. All of which are fundamental building blocks to a website. On top of that, it was necessary for the team to learn the basics of Bootstrap to help build an adaptable and flexible website based on devices used.

Once the basic template and pages of the website is completed, a testing procedure is required to validate that it is functioning correctly. So, Selenium and Mockito were identified as the additional technology that will be used to run tests and simulations for the website. Selenium will test the functionality of the website, whereas Mockito will simulate how the website will work given the presence of a back-end. Fluency in Java will be necessary to use these two technologies effectively.

In the following phase, more knowledge and skills involving back-end development will need to be mastered. Currently, it has been identified that languages such as PHP and Ruby may be necessary on top of Java.

Additionally, readers of this report are expected to understand front web development and the difference with back-end development. Also, the readers will need to know the difference between the languages used for this project and its different uses.

3 Requirements

The problem we are addressing here is how small grocery stores are experiencing losses due to the growth of department stores, supermarkets and other larger chains. These larger stores have multiple advantages over small grocery stores in the form of more competitive prices, access to larger network of distributors and efficient use of retail-oriented technology. Turbodega is designing a system that will lead to the grocery stores gaining parity with these larger stores. To do this, the following goals must be achieved.

1. Reduce costs of restocking supplies

This is done by grouping and sorting orders from multiple grocery stores into orders of an individual type of product. Now an order for a product will be a large order from a group of grocery stores rather than a small order from one grocery store. Hence, the order size for a product will appear as an order from a large supermarket, so distributors can take advantage of the economies of scale to fulfill orders at lower prices.

2. Increase access to more distributors

By placing an order on the system, grocery stores can access the products of all the distributors on the Turbodega network rather than relying on the same distributor. This maximizes efficiency and minimizes costs because distributors will compete to provide the best and cheapest service.

3. Improving the use of technology

Turbodega must be efficient, user-friendly and intuitive, so that grocery stores can accustom themselves with the use of modern-day technology. This encourages them to use and adapt themselves to Turbodega.

To achieve these requirements, the website to be created will require at least four main pages. The pages identified are the registration page, the catalogue page, the bidding page and the bid history page.

1. Registration page

Users must register when they first visit the site. A registration form will include details such as username, password, company details and contact. A key input for registration would be location and address. Due to navigation difficulties in Latin America, it would be best to have a way for users to indicate their location using a map. Location is important to allow filtering of orders based on radius of geographical location (See Figure 5 and Figure 6 of 10.2).

2. Catalogue page

This is where distributors will upload their inventory, in other words, the list of items they will be supplying. The inventory will be sorted based on general categories and it will also help filter orders, so distributors would only see orders for items they are supplying (See Figure 9 of 10.2).

3. Bidding page

Distributors will see the orders that they can bid here and allow the bidding process to take place. The distributors will place bids for any order that they wish to fulfill. All the bids inputted here will be sent to the back-end to process and to determine the lowest best bid. (See Figure 10 od 10.2)

4. Bid History page

Users can see the history of their transactions, where they will be able to see their successful bids, pending bids and unsuccessful bids. The distributors will have to fulfill their successful bids (See Figure 11 of 10.2).

Possible constraints are based on the bidding process where there is a bidding time frame and how a bid cannot be changed after it closes. In addition, a sheet will be uploaded with stock keeping units (skus) to automatically update the catalogue page, hence it is possible that a sku is not read and would require manual inputting. It is also possible that distributors deliver the goods late, which will become a problem for store owners. Lastly, making the user interface intuitive for the first-time users keeping our target clients in mind is also a constraint.

4 Design and Results

4.1 Early design decisions

Initially, we were expecting to work on a mobile solution for Turbodega. The idea was that the app built would allow owners of small groceries to pool their orders together to access the logistical and economies of scale available to supermarkets. We started looking into tools that were suited for the design of mobile applications such as Android Studio and researching mobile oriented UI/UX. However, there was another team that joined the project, so after consulting with the founders and Professor Varro, we decided to work on building the web platform that will be used by the distributors.

Once that decision was made, following an agile development approach, we regularly met with the founders and Professor Varro, continuously listening to their feedback on how the website should look like and adjusting accordingly.

From our first discussions, we understood that the founders wanted a login page, a page with a catalogue of products from which the distributors could select the products they would be interested in, a page where all the possible bids available are shown and a history of the bids made by the distributor using the application. We divided the workload between us and built a minimum viable product with mostly HTML and CSS.

After the minimum viable product was completed and presented to the founders, since the website was very bare bones in terms of presentation, we agreed to put all our focus on giving the website a modern look and feel instead of adding more functionality. Indeed, getting the UI right would be critical to getting distributors to use the website, thus we decided to heavily incorporate Bootstrap into our application's code-base.

In a subsequent meeting, after presenting the website redesigned with Bootstrap, the founders approved of most of the updated web pages but raised a few issues about the catalogue page. Specifically, we had it built such that once the distributor would select a category of the products he had in inventory, all the products in that category would show up. Since we were only working with a small data sample, it did not occur to us right away that at scale, this could quickly become problematic. After further discussions, we realized that letting the distributors choose the products they cover would be better done with them uploading an Excel/CSV file with their inventory and then add or remove individual products afterwards. Hence, we decided to also include a search bar to ease the latter. Professor Varro also gave us a lot of pointers on how to make the UI more intuitive. For instance, he suggested to place certain features on different tabs and to remove certain elements if it took unnecessary screen space or if it did not provide much utility to the user (see Appendix 10.1).

4.2 Results obtained this semester

We have mostly completed building the front-end of the Turbodega web application. Specifically, the login page, the active bids page and the bid history page are all done from a front-end perspective. We only must add a file upload option on the catalogue page for distributors to upload their inventory from an Excel/CSV file when they start using the website.

Since there is no back-end yet, a user can visit the registration page and enter their personal information, but it will not be saved. Similarly, visiting the catalogue page, the active bids page or the bid history page will only yield static content for now.

Concerning the architecture, we are aiming for a design where the registration, the user-specific catalogue, the active bids and the bid history pages are on the presentation tier. Back-end-wise, we intend to compare user credentials with those stored in our database to verify users' identity when they login to the website. Also, we need to retrieve their custom catalogue, their bid history and regularly pull a list of available bids sorted by volume and price to display to the users (see Appendix 10.3 for a table).

4.3 Suitability of the tools used so far

For version control, we used GitHub, whose use is ubiquitous in the industry. It allowed us to keep track of the contributions of each team member while facilitating the distribution of the workload among us. This enabled us to work in parallel where team members would work in their own branches. Once the feature they were tasked to work on were completed, they would merge it into the master repository.

For building the actual application, as mentioned above, we started by writing HTML and CSS from scratch. Eventually, we decided to a Bootstrap framework to be more efficient with our time dedicated to development. Bootstrap is advantageous as it significantly reduces the time spent on tinkering with CSS to get a modern look and feel for the UI. It also makes the website dynamically adjust its layout according to the resolution of the browser accessing it. Moreover, it is also a very well-known framework, so future developers working on this project will be able to easily pick up and extend the functionality of the application.

Concerning the testing tools that we used so far, we looked at Selenium and Mockito, which are both popular testing frameworks used in web development. Selenium allows us to automatically test for routine usage of the UI by simulating a virtual user. Selenium can perform any sort of automated interaction but was originally intended and is primarily used for automated web application testing. For example, by using Selenium, we can test the login process that involves the user entering a username and password in a form and then clicking a sign in button. Then, the website should display the login response message and lead the user to the home page. Since we do not have a back-end built yet, we investigated using Mockito to mock calls to the back-end, so that our code's architecture will easily interface with the back-end once we start working on it. Mockito is an open source testing framework for Java that allows developers to create mock objects for use in automated unit tests. It is often used in the Test-Driven Development and Behavior Driven Development approaches. Using Mockito mock objects, developers can mock external dependencies and ensure their code interacts with it in an expected manner. Before we can start creating mock objects using Mockito, we first need a Java class for that can mock the behavior. Mockito can verify a method is invoked exactly once or a few times and it can verify methods that are invoked orders. However, we still need to progress on this area since we had to focus most of our efforts in getting the UI completed in the later stages of the project.

5 Plan for Next Semester

5.1 Frontend Development

While the UI is mostly done, we still have to add a file upload option on the catalogue page for first-time users of the application. Also, while this is not critical, we have also thought about combining React with Bootstrap for making the application more modular. This would enable the founders to add more functionality with less development time.

5.2 Backend Development

While this semester we are focusing more on building the website, specifically on the user interface, for the next semester we will be working more on the website's back-end and the testing. Every team member will be responsible for their respective web pages' back-end and testing. For the back-end purposes, we decided on using Node.js as it is an open source server environment that is free, and it also runs on various platforms like Windows and Mac OS X. To have a great website, we should be able to generate dynamic page content and it must also be able to collect form data. Node.js can be used as a tool to carry out these tasks.

For Turbodega website, there are several web pages that require the access of servers that can store and serve data from a database. For example, login and registration pages need a database that will take care of the distributors' information. When new users register, Node.js will be used to collect their information and validate the email address and password that they provide and store this information into a database. When existing users sign-in, Node.js will take care of the authentication process. It is also important for us to have a server that has session management as its core of the authentication system. This will allow a user to stay logged into the website and they will not have to re-enter their credentials before viewing each page.

The most secure way to handle user sessions is via server-side cookies. Node.js has an express-session library that allows us to create and manage server-side cookies. In addition, this library handles a lot of work behind the scenes such as creating secure, cryptographically signed cookies, so we can store data in a user's browser. Cryptographic signing is a technique that allows the server to tell whether or not a user has tried to modify their cookies to impersonate someone else. Furthermore, the library gives a simple API for creating and removing cookies that we can tweak and configure cookie settings based on what we need to do.

Also, our website consists of a catalogue page. This page is for the distributors to keep track of their inventory and they will only be notified on auctions that are related to the goods they supply. We will use Node.js to allow distributors to upload excel files containing the products including their skus and brands. We should be able to populate the list of products according to their categories by using the files uploaded. The products in their inventory will be stored in a database and this must correspond to the list of orders they will see at the active bids page.

At the active bids page, the distributors should be able to set the price of their goods. A real-time auction website may seem complicated. The website requires two things; a real-time display of the incoming flow of bids to all distributors and storage of the flow of bids from the beginning of the auction. Node.js on the server side has the built-in feature needed. The real-time display of bids let a participating distributor at a browser to submit a bid. Unlike the normal auction websites, users of Turbodega will not be informed on the prices that others bid for. They will only be notified whether they win the bids or not after forty-eight hours of the auction session. Node.js allows us to use HTML and JavaScript to create 'Auction Server' and 'Auction Client'. The distributor can insert the bid amount at an input box and then press the submit button. Once a user submits a bid, it will transmit it to the auction server. All the bids will go through the auction server and these bids will be stored in a database. After forty-eight hours, the lowest bid price will be chosen. The distributors will be notified, and they can view the price they bid for and their bid status at the bid history page. It is important for us to have a database that can link the database for active bids page and bid history page. This functionality will help distributors predict the future bid they want to set.

Node.js has the library and functionality that we need for the back-end development. Still, we must manage our own servers and write back-end functionality. Dealing with server management can be a little difficult, hence we need to keep our option open in terms of the technology tools used. Another tool that we considered for back-end development is Firebase as it provides almost similar functionality as Node.js with an advantage of not having the need to manage any server at all. Nonetheless, in terms of scalability aspect, Firebase tends to be a less cost-efficient option as compared to Node.js. However, since our application is mainly business-to-business oriented, scalability might not affect our choice as much.

5.3 Testing

For testing purposes, we are planning to use Mockito and Selenium, just like we did for this semester. In any case, when unit-testing a class, we want to test only its functionality and not that of its dependencies. To achieve this, we need to provide a mock to the object under test, a replacement that we can control for that dependency. This way we can force extreme return values, exception throwing or simply reduce time-consuming methods to a fixed return value and reduce test execution time. We plan on continuing to use Mockito for testing because Mockito focuses on having a very simple and clean API. This allows ease of use and a simple way to get started. It allows users to use the same techniques when mocking classes or interfaces as there is only one kind of mock and one way of creating mocks. Users can easily track failed verification in tests and these exceptions lead the user to the actual point of interaction in the code. The stack traces will always be clean, and verification can be flexible in the order it is presented ensuring the most important verification can be shown first. Although Mockito is a mature framework, it lacks the ability to mock static or private methods. This is because Mockito prefers object orientation and dependency injection over static, procedural code that is hard to understand and change. To solve this limitation of Mockito, we might consider using PowerMock, an extension to Mockito that allows for testing of static methods when needed.

Another testing tool that we want to continue using is Selenium. Selenium is a powerful web testing framework that has the capability of automating web applications for testing purposes. Selenium is also known for its ability to automate web-based administration tasks. Selenium comes in two parts; the Selenium WebDriver for creating powerful, browser-based automation suites and test, meanwhile Selenium IDE is used for creating quick bug reproduction scripts. We chose to stick to this technology because of its multiple language support and it is also not restricted by platform barrier. Selenium gives testers the ability to test across major web browsers to provide a smooth user experience for users across different browsers. Besides browsers, Selenium can also be used to test on mobile such as Android, iOS, Windows and Blackberry applications.

6 Impact on Society and the Environment

The target user group for this project is the grocery store operators and their suppliers. At status quo, they are operating at levels that are inefficient and creates waste. Therefore, this project will hopefully change this and produce a positive impact on the related society and environment by the end of its implementation. Its impact throughout development may be slightly limited or difficult to observe due to its nature of being a software-oriented project. This is because development will solely involve the use of personal computers and multiple different software.

However, it is still possible to be environmentally conscious throughout development. For example, it is possible to minimize usage and waste of electricity while developing on the computers. Given the context and the amount of usage, it seems the energy used will be negligible. Nevertheless, the little things do add up and the team does its part in contributing in energy management.

It is necessary to be aware of the eventual impact of the project on the environment given that it will be implemented successfully. If it does, then there will be less waste of resources. Distributors who use the website will be able to deliver higher amounts of goods using less resources. For example, delivery trucks will be able to carry more goods to the optimal delivery point, hence not wasting time and fuel going back and forth to different locations. Grocery store operators will also be less prone to overestimate their orders since they will be able to better keep track of what they need when ordering their goods.

The risks involved in this project is minimal. Yet, like all software projects, it is important to be able to keep track of any changes to the master code and keep backups of everything important. Aside from that it is important to be wary of accessing the internet for references or guides or downloading libraries in the case of potential viruses. The website must also be developed to be secure when keeping and storing the users' information.

Besides that, once the website is fully functional and used by the distributors and grocery store owners, it will hopefully significantly grow the economies of the users. The profits of the grocery store owner will grow and will be able to compete with larger supermarkets and therefore improve their quality of life. This will be able to rejuvenate the small grocery store industry leading to possibly newer and more accessible grocery stores opening at more isolated locations. Thus, people living near grocery stores will no longer need to go to large supermarkets for cheaper products that may otherwise be inaccessible to them.

As mentioned before, most of the impact due to this project will only be observed and properly evaluated at the end of it once it is implemented. Though, it is still important to acknowledge and be aware of it while running the project.

7 Report on Teamwork

We are a team of four people where each team member contributed to the project equally. Our plan for sharing the work as a team is to divide the workload based on personal interest and individual strengths. Table 1 highlights the major responsibilities each member had.

Team Member	Work Description
Yahya Azami	• Building the active bids page of the website • Responsible for various sections of the progress reports • Responsible for setting up the test environment
Aliah Mohd Nazarudin	• Created the overall website layout and theme • Responsible for the registration and login page of the website • Responsible for various sections of the progress reports • Responsible for putting the presentation slides together
Nabil Ersyad Noor Eddie Putera	• Responsible for the bid history page • Responsible for various sections of the progress reports
Thusha Sivapatharajah	• Created the initial website with pages linked together • Responsible for the catalogue page of the website • Responsible for various sections of the progress reports

Table 1: Work Breakdown

Since we are creating a website containing many pages, each member was responsible for a web page as mentioned in Table 1. Though these were our main tasks, each member helped with other responsibilities. For instance, Yahya worked on setting up test suites using Selenium, so that the rest of us can learn from it to run our own tests. Whereas, Thusha combined everyone's individual pages together to have the initial setup of our website in place and have an easily navigable site. Meanwhile, Aliah with the help of Nabil used a Bootstrap theme to make our pages more presentable and responsive. In addition, each member took turns updating the minute's document and had individual parts to cover for every report that is to be submitted.

We were able to collaborate as a team since we helped each other out and covered each other when one needed help. Communication is managed between team members by scheduling weekly meetings every Monday evenings in Armstrong 3rd floor and for other inquiries, Facebook messenger was our source of communication. Furthermore, to collaborate our work, all written work was managed on Google Drive and Overleaf for review and edit. GitHub was used for setting up our website and finalizing every work.

As a team, we have progressed from being a group that did not have background skills in web development to now creating our own active website. This was our main difficulty since the project was software based. To overcome this difficulty, we took time to review tutorials online and tested these skills by first creating

our own blog pages on GitHub to get familiar with HTML/CSS. Then, we attempted to make our assigned pages with Bootstrap to make our website more appealing and responsive. Now, for the remainder of the project, we will need to incorporate React with Bootstrap, set up a proper test suite through Selenium and Mockito to ease development of future iterations and get familiar with Node.js for our back-end. To overcome this, we will have to review some more tutorials online regarding these topics. Other than that, there were not any difficulties encountered in this project so far and hopefully the remainder of the project will be the same.

8 Conclusion

We completed the front-end part of our web application with the only thing missing being a file upload option module on the catalogue page to let distributors upload an Excel/CSV file with the list of products they have in stock. What remains for us to do next semester is implement the business logic on the back-end using Node.js and build a full testing suite to make extending the application easier in the future. We are also considering rewriting parts of the front-end using React to increase modularity.

For a team of developers with little to no experience with building web applications, we have learned a lot so far about the tech stack used for front-end development, but also about intuitive UI design and the logistics of undertaking a big development project in a team. Moreover, due to the agile development approach we have been using, we learned substantially about communicating and working directly with clients. In that respect, we realized that we often must adjust the terminology or to include examples when discussing what we're planning to build or request that our clients illustrate what they have in mind so that we're all on the same wavelength.

9 References

[1] E. Torres, 4e, Camino al Progreso Shopkeepers with business and social focus, SABMiller, 2014.

[2] CEPAL, Portal de Informacion estadistica (http://estadisticas.cepal.org/)

[3] 4e Project, 2014. With information of 3614 tiendas

10 Appendix

10.1 Sample UI changes

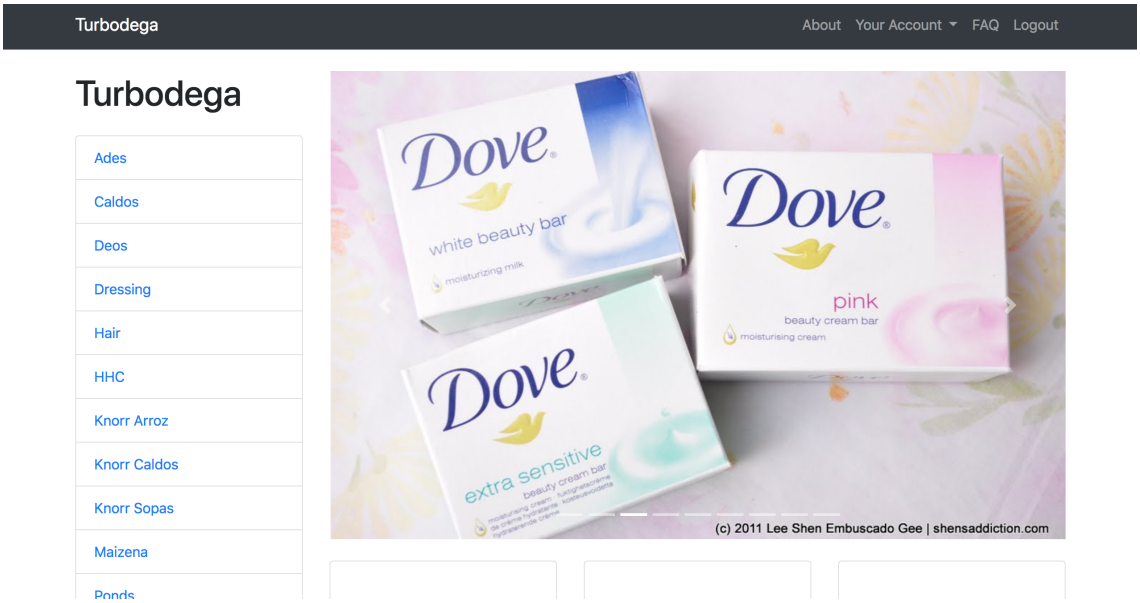

Figure 1: Example of the catalogue webpage before making the UI more intuitive

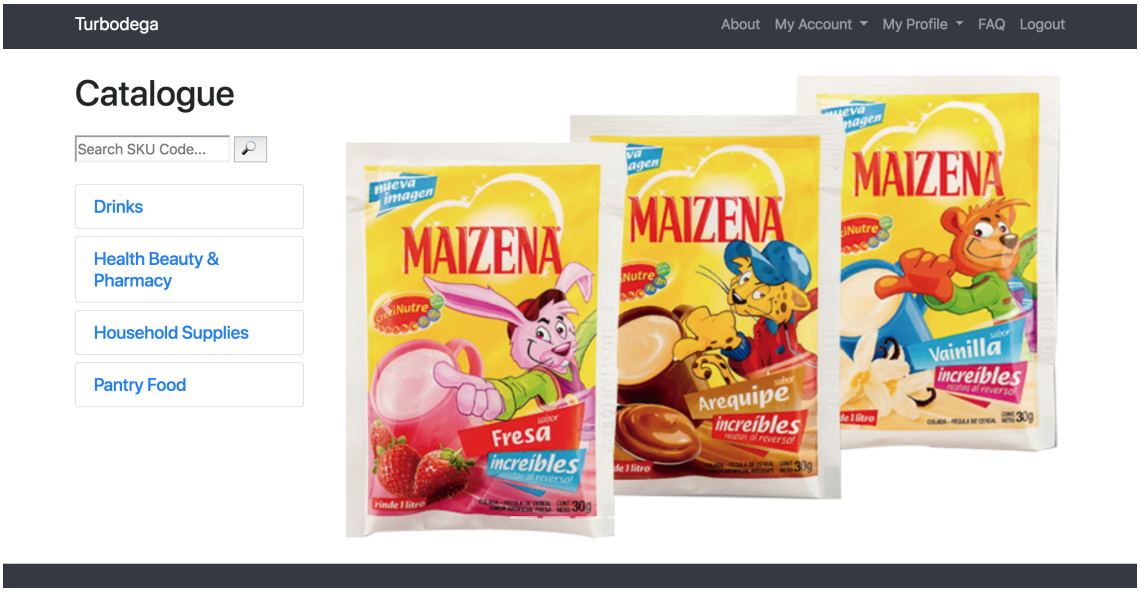

Figure 2: The slideshow appears only if the customer doesn't select any categories and disappears right after, whereas it was always showing up on the webpage's iteration. Also, another tab was created for updating personal information called "My Profile" as opposed to having it included in the "My Account" tab.

10.2 Webpages

Link to the website: https://turbodevg3.github.io/turbodega/

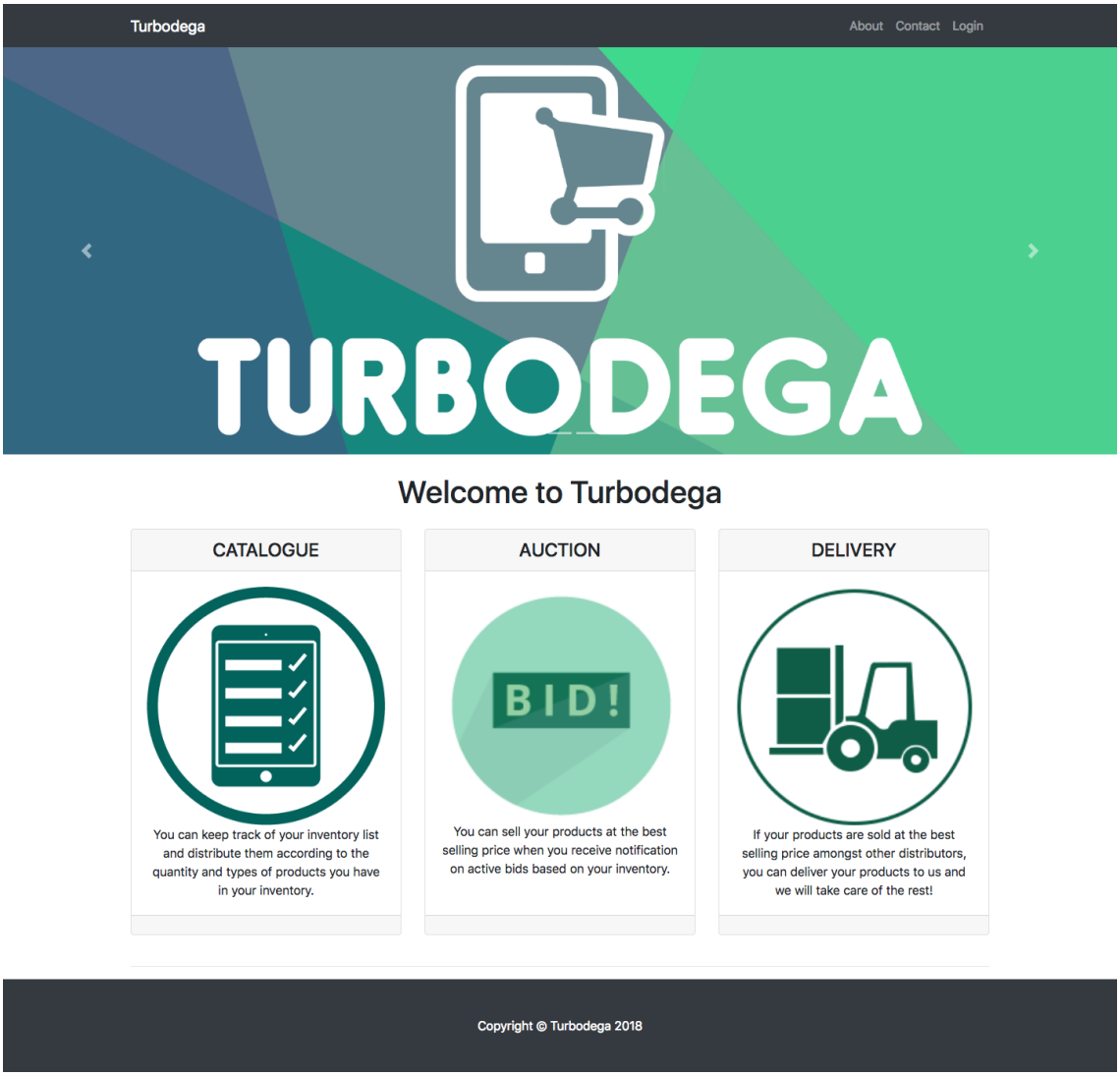

Figure 3: Landing Page

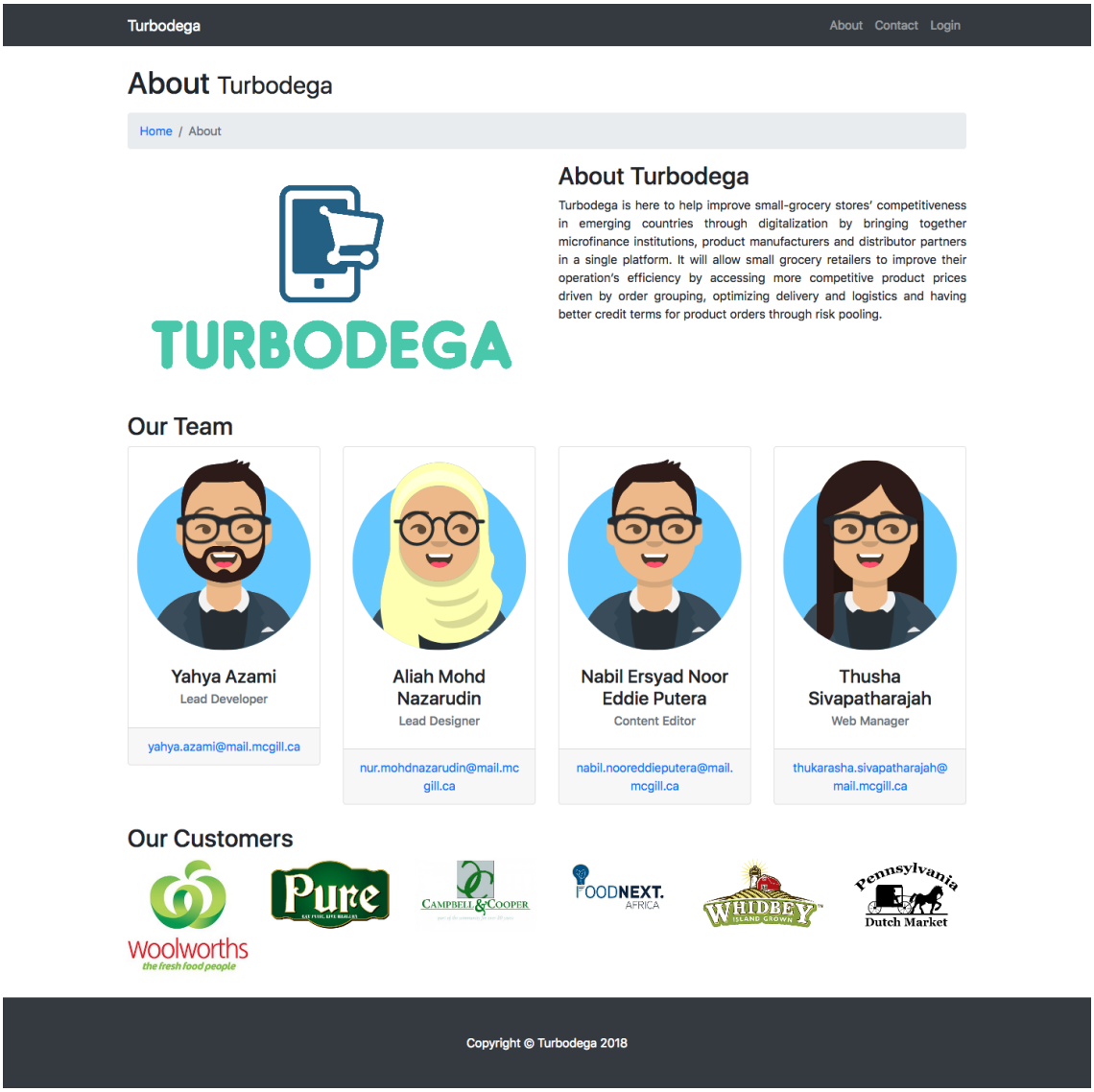

Figure 4: About Page

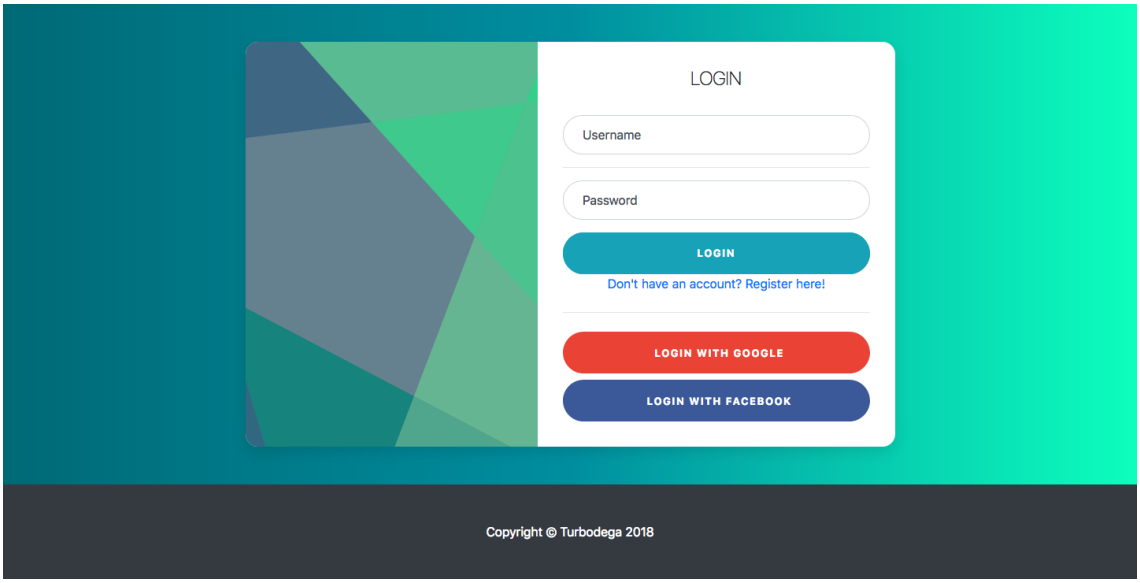

Figure 5: Login Page

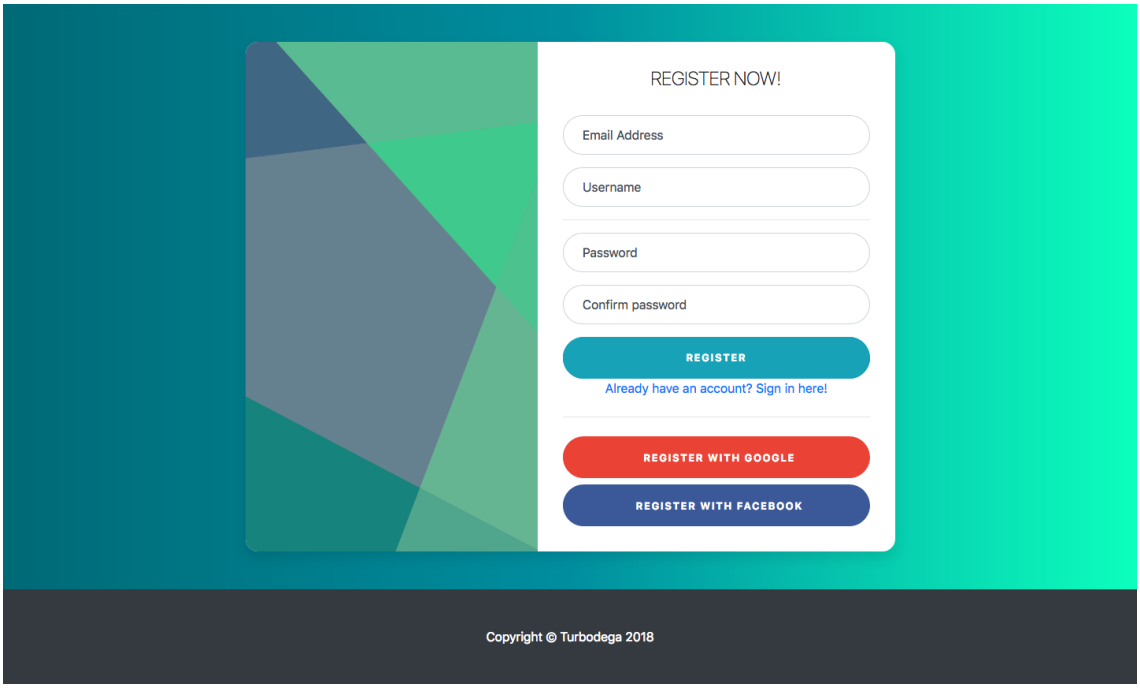

Figure 6: Register Page

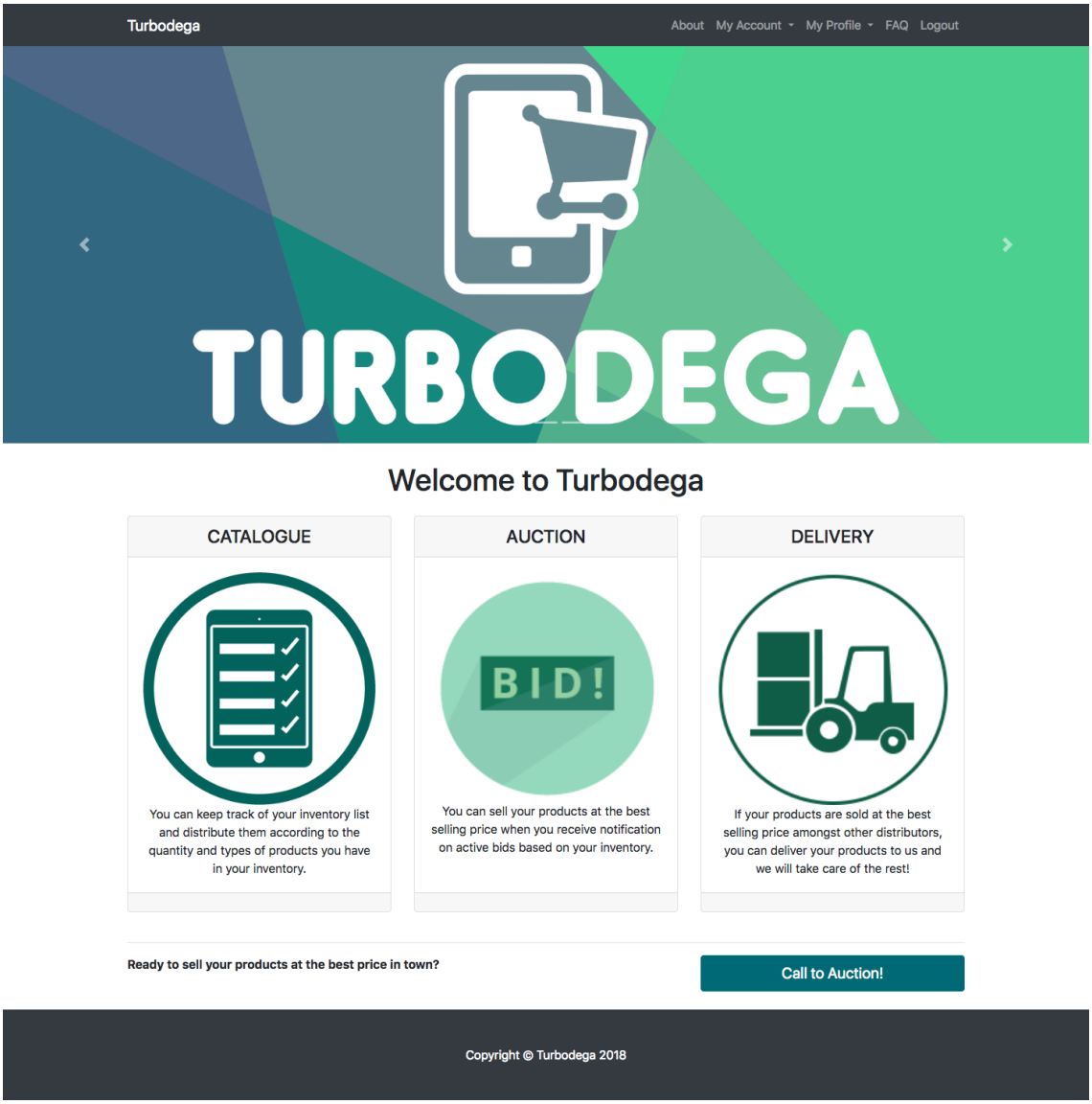

Figure 7: Home Page

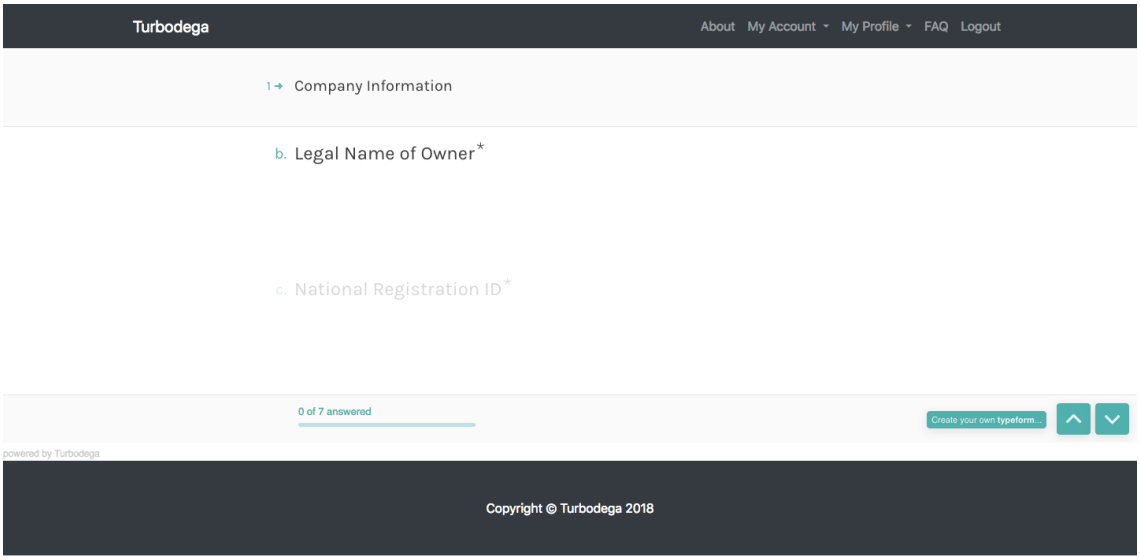

Figure 8: Account Page

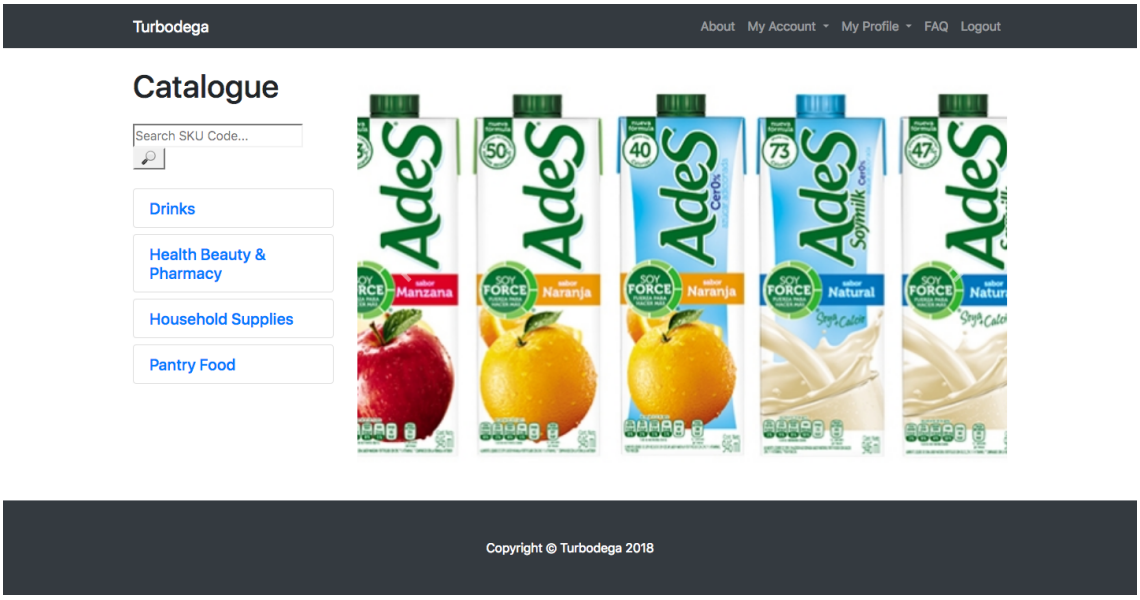

Figure 9: Catalogue Page

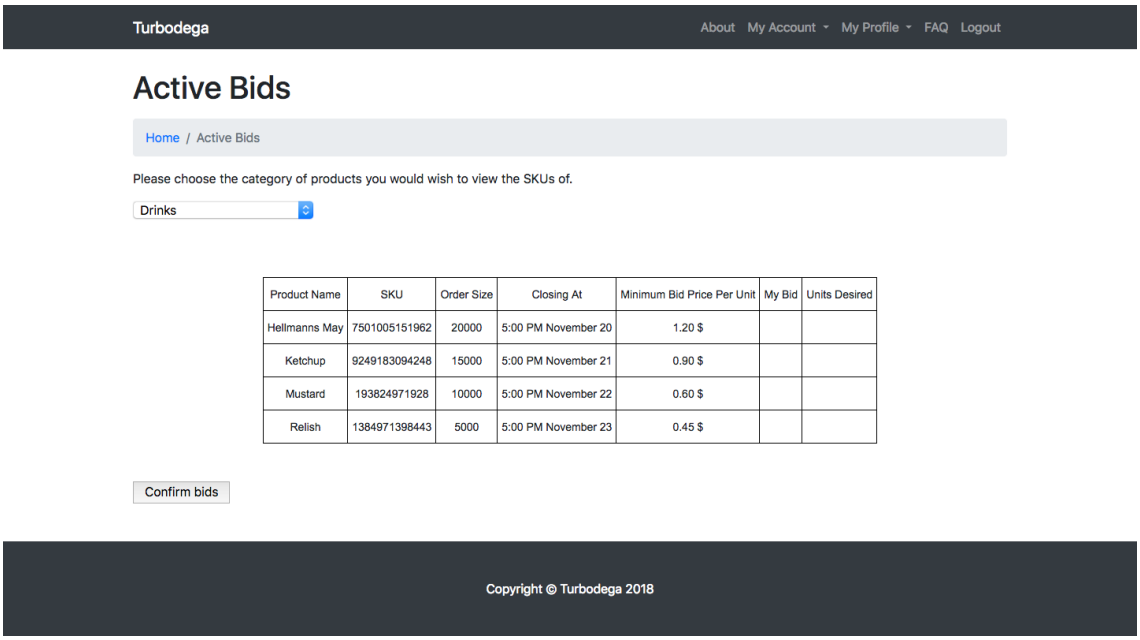

Figure 10: Active Bid Page

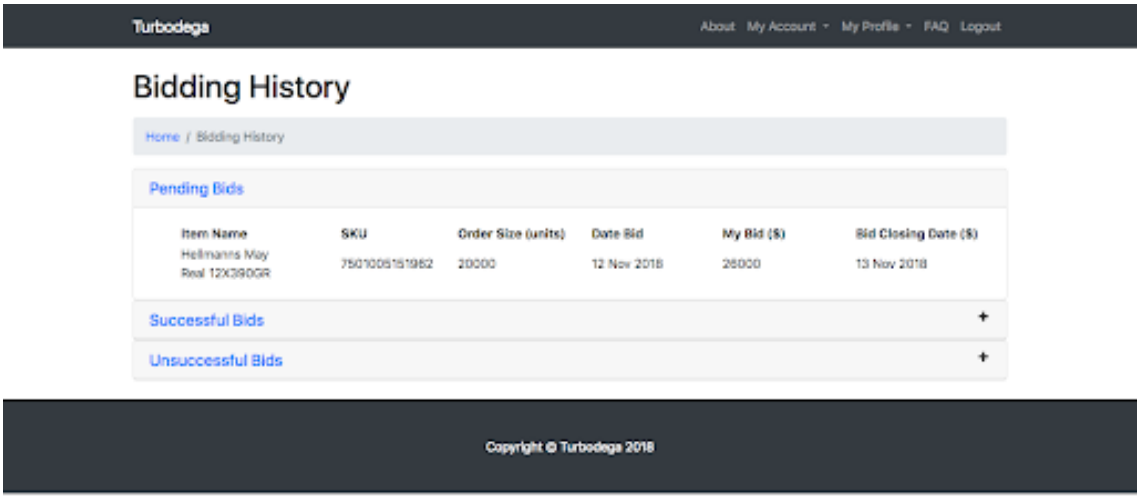

Figure 11: Bid History Page

10.3 Architecture

WEBPAGES	Login	Catalogue	Active Bids	Bid History
PRESENTATION TIER	Registration	Build user-specific catalogue	Display available bids	Display user bidding history
LOGICAL TIER	Compare user credentials to verify identity	Retrieve product information, store user inventory	Fetch available bids, sort available bids by volume/price	Retrieve user bidding information
DATA TIER	Database			

Figure 12: Website's planned architecture